SOUND SYNTHESIS FOR PHYSICS-BASED COMPUTER ANIMATION

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Jeffrey Chadwick

August 2013

SOUND SYNTHESIS FOR PHYSICS-BASED COMPUTER ANIMATION

Jeffrey Chadwick, Ph.D.

Cornell University 2013

In this thesis, we explore the problem of synthesizing realistic soundtracks for physics-based computer animations. While the problem of producing realistic animations of physical phenomena has received much attention over the last few decades, comparatively little attention has been devoted to the problem of generating synchronized soundtracks for these simulations. Recent work on sound synthesis in the computer graphics community has largely focused on producing sound for simple, rigid-body animations. While these methods have been successful for certain scenes, the range of examples for which they produce convincing results is quite limited. In this thesis, we introduce a variety of new sound synthesis algorithms suitable for generating physics-based animation soundtracks. We demonstrate synthesis results on a variety of animated scenes for which prior methods are incapable of producing plausible sounds.

First, we introduce a new algorithm for synthesizing sound due to nonlinear vibrations in thin shell structures. Our contributions include a new thin shell-based dimensional model reduction approach for efficiently simulating thin shell vibrations. We also provide a novel data-driven model for acoustic transfer due to vibrating objects, allowing for very fast sound synthesis once object vibrations are known. We find that this sound synthesis method produces significantly more realistic results than prior rigid-body sound synthesis algorithms for a variety of familiar objects.

Next, we further address the limitations of prior sound synthesis techniques by introducing a new method for synthesizing rigid-body *acceleration noise* – sound produced when an object experiences rapid rigid-body acceleration. We develop an effi-

cient impulse-based model for synthesizing sound due to arbitrary rigid-body accelerations and build a system for modeling plausible rigid-body accelerations due to contact events in a standard rigid-body dynamics solver. This allows us to efficiently recover acceleration sound using data readily available from rigid-body simulations. Our results demonstrate that our method significantly improves upon the results available when using traditional rigid-body sound synthesis with no acceleration noise modeling. We also introduce a scalable proxy model which provides us with a practical method for synthesizing acceleration sound from scenes with hundreds to thousands of unique objects. This allows us to produce substantially improved sound results for phenomena such as rigid-body fracture.

Finally, we also consider sound from other, non-rigid phenomena; specifically, sound from physics-based animations of fire. We propose a hybrid sound synthesis algorithm combining physics-based and data-driven approaches. Our method produces plausible results for a variety of fire animations. Moreover, our use of data-driven synthesis grants users of our method a degree of artistic control.

BIOGRAPHICAL SKETCH

Jeff Chadwick was born in Mississauga, Ontario, Canada. He attended secondary school at St. Thomas Aquinas high school in Oakville, Ontario, where he developed a keen interest in computing after taking several courses on computer programming. After high school, he attending the University of Waterloo to pursue a double major in Applied Mathematics and Computer Science. While at Waterloo, he participated in a number of industrial and academic internships. These included working as a software developer for Open Text Corporation, and as a research assistant for Professor Sue Ann Campbell in Waterloo's Department of Applied Mathematics. Jeff also spent several semesters as an intern software developer at Side Effects Software in Toronto, Ontario. It was there that Jeff was introduced to computer graphics and physical simulation, two topics that would form the basis of his Ph.D research. The experience Jeff gained in graphics and simulation while working at Side Effects ultimately prompted him to pursue graduate research in these topics. Jeff has been studying for his doctorate degree in Computer Science at Cornell University since 2007.

This thesis is dedicated to my parents, Kevin and Carole, whose support has made this possible.

ACKNOWLEDGEMENTS

First and foremost, thank you to my adviser Professor Doug James, whose advice and enthusiasm has made this research possible. During my time at Cornell I also had the opportunity to work closely with Professor David Bindel, who was a valuable source of research insight and inspiration. I would also like to thank my other collaborators – Steven An, Ted Kim, Steve Marschner and Changxi Zheng – and my other Ph.D. committee members, Professors Charles Van Loan and Daniel Huttenlocher. In particular, thank you to Steven An, who was also my roommate for several years and provided a constant source of advice and encouragement.

The time I spent at Side Effects Software as an undergraduate co-op student allowed me to develop my interests in computer graphics and physics simulation, even before beginning work on my Ph.D. I would like to thank the numerous talented individuals whom I had the chance to work with at Side Effects, particularly Jeff Lait, who introduced me to many topics in physics simulation that were important parts of my Ph.D. research.

Finally, thank you to all of my friends and family for their ongoing support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Over the last few decades, the field of computer graphics has been tremendously successful in its pursuit of algorithms for synthesizing realistic visual images. Sophisticated physically based rendering techniques have enabled the production of images nearly indistinguishable from reality. In addition to work on image rendering, a great deal of research in the computer graphics community has focused on methods for synthesizing physically plausible animations of real-life phenomena. Enormous effort has been invested in to developing methods for animating rigid-body dynamics [6, 58], cloth and other deformable bodies [7, 125, 126], fracturing rigid and deformable bodies [5], smoke [120, 42], fire [98, 64, 65] and splashing fluids [41, 95]. These techniques can produce highly realistic animations of natural phenomena. However, they are in general incapable of producing additional, multisensory feedback to accompany their visual results. In particular, comparatively little effort has been invested in to developing automated methods for generating realistic soundtracks to accompany physics-based animations.

Sound is essential to the process of producing realistic computer animations. Indeed, sound production is an indispensable part of both the film and video game industries. Sound effects for these mediums have traditionally been authored using pre-recorded audio clips produced by *foley artists*. These artists use a variety of creative methods to record and edit audio in order to produce sound effects suitable for film or game production. Unfortunately, this method is both time-consuming and labor-intensive. Moreover, for animated scenes produced with physics-based solvers this method is discarding valuable physical information from the scene which may be helpful for producing a realistic soundtrack. Beyond applications in the entertainment industry, building realistic virtual environments hinges upon more than just compelling visual feedback. A convincing

virtual environment must also provide realistic multisensory feedback, including sound.

The process of synthesizing sound for a computer animation can be roughly divided in to two components; *sound generation* and *sound propagation*. Sound generation refers to the mechanism by which a particular phenomenon produces sound. For instance, we must study how a cymbal vibrates when it is struck with a drum stick to gain an understanding of how the cymbal produces sound. Sound propagation refers to the process by which sound waves are produced by some physical phenomena. This may also include *environmental acoustics*; that is, modeling the interactions of sound waves with a physical space to create echoes and reverberations. While this problem has received much attention in the acoustics research community (see, e.g., [51, 50, 24, 105]), it is not addressed in this work. In this thesis, we focus on the sound generation problem for a variety of physical phenomena. We also consider how sound propagates from these phenomena in order to build plausible models for what a listener placed in a virtual scene might hear.

Within the graphics community, the problem of synthesizing sound for rigid-body physics animations has received a fair amount of attention [100, 129, 12, 70]. This problem admits a relatively simple physical description (see §2.2.1) which allows for efficient physics-based simulation of the sound produced by these animations. Existing rigid-body sound synthesis methods have been successfully applied to a number of objects (for example, the ceramic plates in Figure 1.1). However, they tend to produce unconvincing results for a wide variety rigid or nearly rigid objects (see Figures 1.2. and 1.3). Moreover, while rigid-body objects are an important sound source, there are numerous recognizable sound sources which do not fall under this category. These include sound from highly deformable objects like cloth, aerodynamic noise (e.g., wind blowing through a small opening), and combustion sounds (e.g., the sound from waving

Figure 1.1: Popular Rigid-body sound synthesis methods in the graphics community are capable of producing compelling sound tracks for simulations such as this scene in which several ceramic plates fall on a wooden table.

a flaming torch through the air). Sound synthesis for these complex phenomena has received little attention in the computer graphics community.

In this thesis, we build upon prior in rigid-body sound synthesis by introducing new algorithms to address some of the limitations discussed above. We also introduce a method for synthesizing plausible sounds synchronized with physically based animations of fire. Our contributions are organized as follows:

Chapter 2 provides a basic description of the existing rigid-body sound synthesis pipeline that our contributions will build upon. This includes an overview of the physics of object vibration and a short introduction to the physics of sound propagation.

Chapter 3 introduces a method for synthesizing sound due to nonlinear vibrations in nearly rigid thin shell objects (e.g., cymbals, water bottles and metal trash cans). By incorporating nonlinear effects in to the rigid sound synthesis pipeline, we are able to

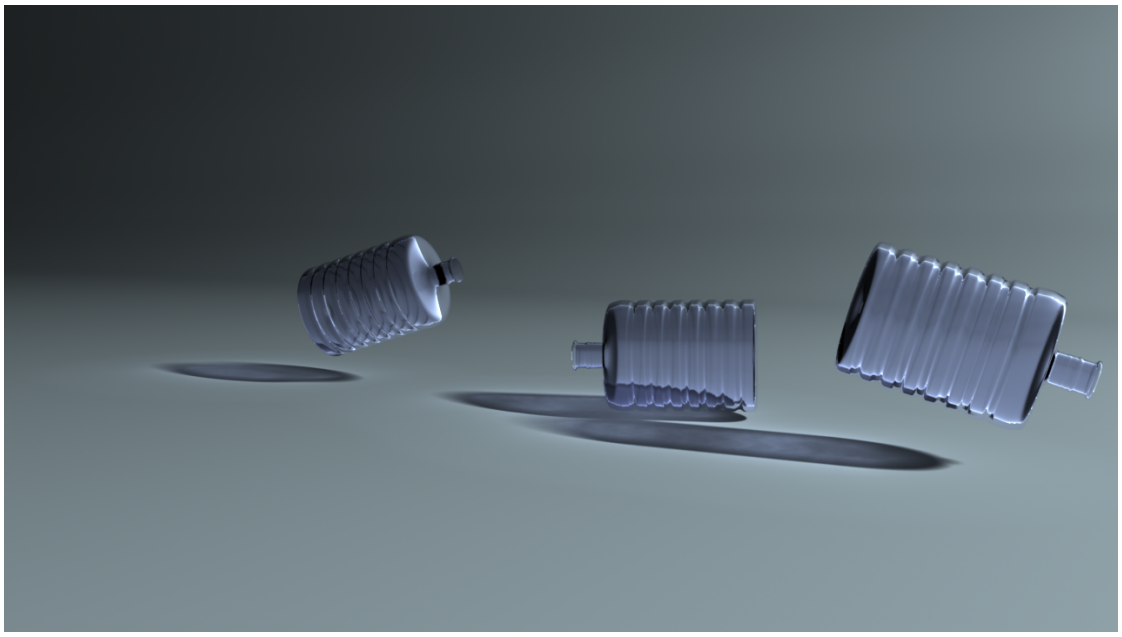


Figure 1.2: Objects like these plastic water bottles tend to experience nonlinear vibrations when they collide with each other. Existing rigid-body sound synthesis algorithms fail to resolve the nonlinear physics of these objects and produce unsatisfactory results.

synthesize results that are much more convincing than those available with prior methods (see Figure 1.2).

Chapter 4 describes our algorithm for resolving rigid-body *acceleration noise*; that is, sound produced when objects experience abrupt rigid acceleration. This phenomena has previously not been modeled in rigid-body sound pipelines, resulting in inaccurate sounds for a variety of common objects. Our method efficiently resolves acceleration sound and allows us to substantially improve upon prior results.

Chapter 5 builds on the methods of Chapter 4. We introduce a more scalable version of our acceleration sound pipeline which allows us to synthesize convincing sounds for scenes with hundreds to thousands of objects (see Figure 1.3).

Chapter 6 introduces a new sound synthesis method for physics-based animations of fire. We use a combination of physics-based and data-driven approaches to synthesize

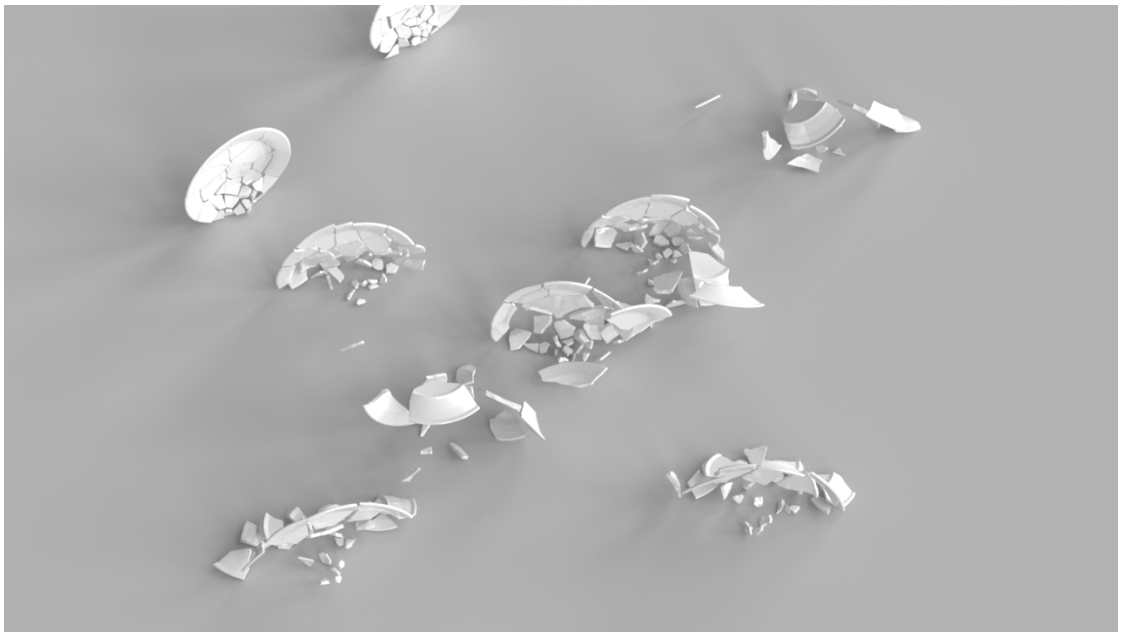


Figure 1.3: Traditional rigid-body sound synthesis methods fail to resolve sounds associated with the purely rigid behavior of objects in a scene. Due to this omission, sounds for a scene like this one tend to sound blurred and indistinct. In fact, several objects in this scene produce no sound at all when existing rigid-body sound synthesis methods are applied.

Figure 1.4: Sound synthesis algorithms are also needed for non-rigid phenomena such as this flaming torch.

sound while avoiding costly fluid simulation with audio-rate time stepping (see Figure 1.4).

Chapter 7 summarizes our contributions and suggests several areas for future research.

CHAPTER 2

BACKGROUND ON SOUND PRODUCTION AND PROPAGATION

In this chapter, we introduce the basic physical models underlying sound production and propagation. We begin with a short introduction to the *acoustic wave equation* – the partial differential equation (PDE) describing sound propagation in a domain – along with a discussion of how sound from either vibrating surfaces or other physical phenomena (e.g., fire, wind, etc.) can be modeled with this equation. Next, we introduce the basic physics of object vibration, as this is relevant to much of the prior work in physics-based sound synthesis for computer graphics.

2.1 Sound Propagation

Sound is produced as the result of pressure fluctuations in a medium (typically air). In particular, pressure fluctuations occurring at frequencies within the range of human hearing (roughly, 20Hz–20kHz) are interpreted by our ears as sound. The way in which these pressure fluctuations propagate through a domain is described by the *acoustic wave equation*.

2.1.1 Acoustic Wave Equation

Sound propagation in a domain Ω is described by the partial differential equation

$$\frac{1}{c^2}\frac{\partial^2 p(\boldsymbol{x},t)}{\partial t^2} - \nabla^2 p(\boldsymbol{x},t) = 0, \quad \boldsymbol{x} \in \Omega, t > 0, \tag{2.1}$$

known as the acoustic wave equation. Here, $p(\boldsymbol{x},t)$ is the pressure referred to earlier, evaluated at point $\boldsymbol{x}$ and time t. Ω is the domain in which sound propagation is modeled

and c denotes the speed of sound, typically taken to be $c = 343$m/s in air at standard temperature and pressure.

Suppose that we wish to consider sound due to vibration of a stationary object O, and let $\Omega_O \subset \mathbb{R}^3$ describe the domain occupied by O. In this case, we let $\Omega = \mathbb{R}^3 \setminus \Omega_O$ and $\partial\Omega = \partial\Omega_O$. That is, Ω is the entire three-dimensional space *outside* of object O. We also define the normal vector $\mathbf{n}(\boldsymbol{x})$ for $\boldsymbol{x} \in \partial\Omega$, and assume that $\mathbf{n}(\boldsymbol{x})$ points in to the domain Ω (or, correspondingly, out of the object O). Furthermore, suppose that, due to the motion of object O, point $\boldsymbol{x}$ on the surface of O has acceleration $\mathbf{a}(\boldsymbol{x}, t)$. Under these conditions, O's effect on the surrounding medium is given by the boundary condition

$$\nabla p(\boldsymbol{x}, t) \cdot \mathbf{n}(\boldsymbol{x}) = -\rho \mathbf{a}(\boldsymbol{x}, t) \cdot \mathbf{n}(\boldsymbol{x}), \quad \boldsymbol{x} \in \partial\Omega,\ t > 0, \tag{2.2}$$

where ρ is the density of the surrounding medium (1.2041 kg/m^3 for air at standard temperature and pressure). Together with a set of initial conditions, (2.1–2.2) describe the time-evolution of acoustic pressure resulting from O's surface motion.

One important property of the acoustic wave equation (2.1) is *superposition* of solutions. Specifically, if $p_1(\boldsymbol{x}, t)$ and $p_2(\boldsymbol{x}, t)$ both solve (2.1) with boundary conditions $\nabla p_1 \cdot \mathbf{n} = b_1(\boldsymbol{x}, t)$ and $\nabla p_2 \cdot \mathbf{n} = b_2(\boldsymbol{x}, t)$, respectively, then $p = p_1 + p_2$ solves (2.1) with boundary condition $\nabla p \cdot \mathbf{n} = b_1 + b_2$.

The discussion above assumes that sound is produced by the motion of a solid surface. This description is valid for a wide variety of scenarios, including sound from rigid-body vibration, cloth or other deformable objects, or even liquid such as water. However, other physical phenomena produce sound even without the presence of any moving surface. This is the case for sound produced by fire or wind, for example. For sound sources such as these, we often treat our domain Ω as being the entire three-dimensional space $\mathbb{R}^3$. Rather than introducing pressure fluctuations via a boundary

condition of the form (2.2), we instead consider the *inhomogeneous* wave equation:

$$\frac{1}{c^2}\frac{\partial^2 p(\boldsymbol{x},t)}{\partial t^2} - \nabla^2 p(\boldsymbol{x},t) = b(\boldsymbol{x},t), \quad \boldsymbol{x} \in \Omega,\ t > 0. \tag{2.3}$$

Here, $b(\boldsymbol{x},t)$ is a source term which depends on the particular sound source being modeled (see Chapter 6 for an in-depth discussion of fire as a sound source).

A general formula for the solution to (2.3) can be derived by making use of the *Green's function* for (2.3); that is, the solution to (2.3) when $b(\boldsymbol{x})$ is given by a delta function source

$$\frac{1}{c^2}\frac{\partial^2 p(\boldsymbol{x},t)}{\partial t^2} - \nabla^2 p(\boldsymbol{x},t) = \delta(\boldsymbol{x}-\mathbf{y})\delta(t-\tau), \quad \boldsymbol{x} \in \Omega,\ t > 0 \tag{2.4}$$

which is given by

$$G(\boldsymbol{x},t;\mathbf{y},\tau) = -\frac{\delta\left(\tau - \left[t - \frac{\|\boldsymbol{x}-\mathbf{y}\|}{c}\right]\right)}{4\pi\|\boldsymbol{x}-\mathbf{y}\|}. \tag{2.5}$$

It then follows that the general solution to (2.3), assuming $\Omega = \mathbb{R}^3$, is

$$p(\boldsymbol{x},t) = \int_0^\infty \int_{\mathbb{R}^3} b(\mathbf{y},\tau)G(\boldsymbol{x},t;\mathbf{y},\tau)d^3\mathbf{y}d\tau = -\frac{1}{4\pi}\int_{\mathbb{R}^3} \frac{b\left(\mathbf{y}, t - \frac{\|\boldsymbol{x}-\mathbf{y}\|}{c}\right)}{\|\boldsymbol{x}-\mathbf{y}\|}d^3\mathbf{y}. \tag{2.6}$$

A variety of numerical solutions are available for solving equations (2.1) and (2.3). Perhaps the simplest is the *Finite Difference method*, in which Ω is discretized using a regular grid. Other popular methods for solving these equations include the *Finite Element* or *Finite Volume* methods.

2.1.2 Helmholtz Equation

In §2.1.1 we considered solving the *time-domain* formulation of the acoustic wave equation. That is, we considered the problem of evaluating the spatially and temporally varying function $p(\boldsymbol{x},t)$. In certain circumstances, it is advantageous to consider the

frequency-domain formulation of this problem. In this formulation, we assume that the solution $p(\boldsymbol{x}, t)$ is *time-harmonic*. That is,

$$p(\boldsymbol{x}, t) = \widetilde{p}(\boldsymbol{x})e^{j\omega t}, \tag{2.7}$$

where ω is a fixed *angular frequency* and j is the imaginary number $j = \sqrt{-1}$. If we are considering sound due to the motion of an object O, then we similarly assume that O's surface accelerations are also time-harmonic: $\mathbf{a}(\boldsymbol{x}, t) = \widetilde{\mathbf{a}}(\boldsymbol{x})e^{j\omega t}$. We refer to $\widetilde{p}$ as the *transfer function* for frequency ω. This function is given by the solution to the Helmholtz equation

$$\nabla^2\widetilde{p}(\boldsymbol{x}) + k^2\widetilde{p}(\boldsymbol{x}) = 0, \quad \boldsymbol{x} \in \Omega \tag{2.8}$$

subject to to the boundary condition

$$\nabla\widetilde{p}(\boldsymbol{x}) \cdot \mathbf{n}(\boldsymbol{x}) = -\rho\widetilde{\mathbf{a}}(\boldsymbol{x}) \cdot \mathbf{n}(\boldsymbol{x}), \quad \boldsymbol{x} \in \partial\Omega \tag{2.9}$$

The parameter $k = \omega/c$ is called the *wave number*. Equations (2.8-2.9) may be solved using a number of standard numerical methods, including the Boundary Element Method. In section §2.2 we will see how transfer functions of the form $\widetilde{p}(\boldsymbol{x})$ are used in the context of rigid-body sound synthesis.

2.2 Rigid-Body Sound Synthesis

In this section we introduce basic techniques for synthesizing sounds from rigid-body objects – objects which do not exhibit visual deformations. This problem has been popular for many years in the graphics community, and many of the contributions discussed later in this thesis build upon the framework described here.

2.2.1 Elastic Vibration

We now describe the basic physical models describing elastic vibration in a solid object. As we discussed in §2.1.1, the surface motion of a solid object produces pressure fluctuations which propagate according to the acoustic wave equation and are interpreted by a listener as sound. The description provided here is meant to act only as a brief introduction. For a more complete overview of the fields of continuum mechanics and elastic vibration we refer the reader to [118, 11].

Solid object vibration may be studied numerically by first discretizing an object in to a set of tetrahedral or hexahedral elements and then applying the Finite Element Method to determine the object's equations of motion [67]. Assuming that an object has been discretized with n vertices, we write the time-dependent position and displacement vectors for these vertices as $\boldsymbol{x}(t), \mathbf{u}(t) \in \mathbb{R}^{3n}$. The general structure of the equations of motion for the time-evolution of $\mathbf{u}$ is given by

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{D}(\mathbf{u}, \dot{\mathbf{u}}) + \mathbf{f}_{int}(\mathbf{u}) = \mathbf{f}_{ext}, \tag{2.10}$$

where $\mathbf{M} \in \mathbb{R}^{3n \times 3n}$ is the object's *mass matrix*, $\mathbf{D}(\mathbf{u}, \dot{\mathbf{u}}) \in \mathbb{R}^{3n}$ are damping forces, $\mathbf{f}_{int}(\mathbf{u}) \in \mathbb{R}^{3n}$ are internal elastic forces, and $\mathbf{f}_{ext} \in \mathbb{R}^{3n}$ are external forces acting on the object.

Most prior work on sound synthesis for computer graphics has assumed that objects are rigid and that deformations $\mathbf{u}$ are small enough that internal forces in the object can be approximated with linear forces $\mathbf{f}_{int}(\mathbf{u}) = \mathbf{K}\mathbf{u}$, where $\mathbf{K}$ is referred to as the *stiffness matrix*. See Chapter §3 for details of a sound synthesis method in which we consider more complex, nonlinear internal forces. A simple damping model known as *Rayleigh damping* is often used to describe $\mathbf{D}(\mathbf{u}, \dot{\mathbf{u}})$. In this model, we have

$$\mathbf{D}(\mathbf{u}, \dot{\mathbf{u}}) = \mathbf{D}\dot{\mathbf{u}} \tag{2.11}$$

where $\mathbf{D}$ is a damping matrix given by

$$\mathbf{D} = \alpha\mathbf{M} + \beta\mathbf{K} \tag{2.12}$$

and α and β are damping parameters The resulting equations of motion for a linear elastic vibrating object are given by

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{D}\dot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{f}_{ext}. \tag{2.13}$$

See [67] for a discussion on how to compute the matrices $\mathbf{M}$ and $\mathbf{K}$ using the Finite Element Method. This formulation has been used previously for sound synthesis in computer graphics in, e.g., [100].

The equations of motion (2.13) can be time-stepped with a variety of numerical integration methods including Euler, Runge-Kutta methods, and implicit/explicit Newmark integrators [67]. High-quality audio synthesis typically requires time-stepping to be carried out at a very high rate (in general, 44 kHz or higher). Time-stepping (2.13) directly at these rates may result in solvers that are prohibitively expensive for animation sound synthesis. Fortunately, (2.13) admits an efficient solution which is particularly attractive for sound synthesis. Specifically, we consider the generalized eigenvalue problem

$$\mathbf{K}\mathbf{u}_i = \omega_i^2\mathbf{M}\mathbf{u}_i \tag{2.14}$$

where ω_i^2 and $\mathbf{u}_i$ are the i^{th} eigenvalue/eigenvector pair, and $\omega_1^2 \leq \ldots \leq \omega_{3n}^2$. If we evaluate only the r smallest eigenvalues and associated eigenvectors, then we can form the matrices $\mathbf{U} = [\,\mathbf{u}_1\,\mathbf{u}_2\,\ldots\,\mathbf{u}_r]$ and $\mathbf{\Lambda} = \text{diag}(\omega_1^2, \ldots, \omega_r^2)$. Without loss of generality, $\mathbf{U}$ and $\mathbf{\Lambda}$ can be chosen so that the columns of $\mathbf{U}$ are *mass orthonormal*; that is,

$$\mathbf{U}^T\mathbf{M}\mathbf{U} = \mathbf{I}_{r\times r} \tag{2.15}$$

$$\mathbf{U}^T\mathbf{K}\mathbf{U} = \mathbf{\Lambda} \tag{2.16}$$

Next, we restrict $\mathbf{u}$ to lie in the r-dimensional subspace spanned by the columns of $\mathbf{U}$. That is, $\mathbf{u} = \mathbf{U}\mathbf{q}$, where $\mathbf{q} \in \mathbb{R}^r$. This transforms the equations of motion (2.13) in to

$$\mathbf{M}\mathbf{U}\ddot{\mathbf{q}} + (\alpha\mathbf{M} + \beta\mathbf{K})\mathbf{U}\dot{\mathbf{q}} + \mathbf{K}\mathbf{U}\mathbf{q} = \mathbf{f}_{ext} \tag{2.17}$$

Finally, by left-multiplying $\mathbf{U}^T$ both sides of (2.17) and making use of (2.15-2.16) we arrive at equations of motion for the vector $\mathbf{q}$:

$$\ddot{\mathbf{q}} + (\alpha\mathbf{I}_{r\times r} + \beta\mathbf{\Lambda})\dot{\mathbf{q}} + \mathbf{\Lambda}\mathbf{q} = \mathbf{U}^T\mathbf{f}_{ext} \tag{2.18}$$

We refer to the vectors $\mathbf{u}_i$ as the *modes* of this system, and to $\mathbf{q}$ as the vector of *modal coordinates*. We also note that (2.18) is a system of uncoupled ordinary differential equations (ODEs) of the form

$$\ddot{q}_i + (\alpha + \beta\omega_i^2)\dot{q}_i + \omega_i^2 q_i = [\mathbf{U}^T\mathbf{f}_{ext}]_i. \tag{2.19}$$

This is the equation for a damped harmonic oscillator. Recall from §2.1.2 that we assume object surface accelerations to be purely time-harmonic. If the object is subjected to an impulse-like external force, then, in the absence of damping, the solution of (2.19) is indeed time-harmonic with angular frequency ω_i. When $\alpha > 0$ or $\beta > 0$, the solution behaves like a damped sinusoid. We acknowledge that the assumption of time-harmonicity certainly does not hold for general, time-varying external forces However, many object's do experience vibrations that are approximately time-harmonic (with damping) when subjected to the impulse-like external forces common during rigid-body simulation. These equations are well-understood and can be evaluated numerically using numerical integrators such as implicit or explicit Newmark [67] or using IIR digital filters [60, 71]. For a mode with angular frequency ω_i, the associated *natural frequency* is $f_i = \omega_i/2\pi$. For the purpose of sound synthesis, we only need to consider vibration frequencies within the range of human hearing. Therefore, r is chosen such that $f_i \leq 20$kHz for $1 \leq i \leq r$. We also discard the first 6 eigenvalue/eigenvector pairs resulting from (2.14), since these correspond to purely rigid translation or rotation. Many

objects can be modeled using no more than a few hundred modes. Moreover, an object's dynamics can be described by a superposition of vibrating mode shapes, whose behaviors can all be resolved independently. This enables very fast evaluation of object vibrations.

2.2.2 Acoustic Radiation from a Rigid-Body Object

Recall from §2.1.2 that for a rigid body with time-harmonic surface accelerations (i.e., with time dependence $e^{j\omega t}$, the acoustic pressure surrounding the object is given by $\widetilde{p}(\boldsymbol{x})e^{j\omega t}$ where $\widetilde{p}$ is the solution of (2.8) with wave number $k = \omega/c$. We observe that solutions to (2.19) are *approximately* time-harmonic. These solutions oscillate with a fixed angular frequency ω_i, but the magnitude of these vibrations is affected by both damping and the external force term $[\mathbf{U}^T\mathbf{f}_{ext}]_i$. For the purpose of building an acoustic radiation model, we assume that that the property of time-harmonic vibration holds. If we interpolate the mode vector $\mathbf{u}_i$ at all points inside of an object, we can consider the continuous displacement field $\mathbf{u}_i(\boldsymbol{x})$ inside of the object. Under the assumption of time-harmonicity, the displacement vector of an object due to the motion of mode i at some point $\boldsymbol{x}$ inside of an object is given by

$$\mathbf{u}_i(\boldsymbol{x})e^{j\omega_i t}. \tag{2.20}$$

Differentiation then implies that the acceleration at point $\boldsymbol{x}$ due to mode i is given by

$$\mathbf{a}_i(\boldsymbol{x}, t) = -\omega_i^2\mathbf{u}_i(\boldsymbol{x})e^{j\omega_i t}. \tag{2.21}$$

Therefore, for mode i, we solve the Helmholtz equation with boundary condition

$$\nabla\widetilde{p}(\boldsymbol{x}) \cdot \mathbf{n}(\boldsymbol{x}) = \rho\omega^2\mathbf{u}_i(\boldsymbol{x}) \cdot \mathbf{n}(\boldsymbol{x}), \quad \boldsymbol{x} \in \partial\Omega \tag{2.22}$$

which is obtained by combining (2.9) and (2.21). We denote this solution $\widetilde{p}_i(\boldsymbol{x})$ and refer to it as the acoustic transfer function for mode i. Mode i's contribution to the time-dependent acoustic pressure field around object O at exterior position $\boldsymbol{x}$ is approximated scaling the time-dependent modal coordinate for mode i with the magnitude of this acoustic transfer function evaluated at point $\boldsymbol{x}$:

$$q_i(t)\left|\widetilde{p}_i(\boldsymbol{x})\right|. \tag{2.23}$$

Finally, we use the superposition principle for the wave equation introduced in §2.1.1 to approximate the total acoustic pressure due to object O at exterior position $\boldsymbol{x}$ as

$$p(\boldsymbol{x},t)=\sum_{i=1}^{r} q_i(t)\left|\widetilde{p}_i(\boldsymbol{x})\right|. \tag{2.24}$$

2.2.3 Linear Modal Sound Synthesis

We now introduce the basic pipeline for the linear modal sound algorithm which has been used previously for computer animation. This model computes sound for objects which are visually rigid; that is, objects that do not visually change shape during simulation.

Precomputation

One of the key advantages of sound synthesis for rigid-body objects compared to other phenomena is the extent to which sound synthesis data can be precomputed, allowing for fast run-time synthesis. Rigid-body modal sound precompution includes the following steps:

1. Discretize the object using Finite Element Analysis to formulate the matrices $\mathbf{M}$ and $\mathbf{K}$ required by the equations of motion (2.13).

2. Solve the linear modal analysis eigenvalue problem (2.14), yielding a set of modes $\mathbf{u}_i$ and associated angular frequencies ω_i.

3. Choose Rayleigh damping parameters α and β. In general, this process is somewhat ad-hoc, and may require manual tuning. However, recent work has suggested techniques for tuning these parameters in a more automated way [108].

4. Precompute representations for the acoustic transfer functions $\widetilde{p}_i$ for each mode $1 \leq i \leq r$. Details of how this process may be carried out are presented in Chapter 3 and have also been discussed in [70, 137].

Sound Evaluation

The precomputed data discussed in §2.2.3 allows us to efficiently evaluate rigid-body sound using data readily-available from a rigid-body solver such as [58]. This process involves the following steps:

1. Run a rigid body simulation and for each object O compute a time-varying vector $\mathbf{f}_{ext}(t)$ of external forces acting on the object. For sound synthesis, contact forces between colliding objects are the most common forces considered here.

2. Solve the equations of motion (2.19) for object O's mode set. The ODE (2.19) can be solved using a variety of numerical integrators (e.g., implicit Newmark) or using an IIR filter [71]. This yields a time-varying vector of modal coordinates $\mathbf{q}(t)$.

3. Given the position of a virtual microphone $\boldsymbol{x}(t)$, transform this position from world space in to the rest frame of object O. This gives a time-varying microphone position $\widetilde{\boldsymbol{x}}(t)$ relative to O's rest frame.

4. Using the modal coordinates $\mathbf{q}(t)$ computed in step 2, evaluate the total sound pressure at the listener's position $\widetilde{\boldsymbol{x}}(t)$ using (2.24).

CHAPTER 3

HARMONIC SHELLS

In this chapter, we consider sound from a class of nearly rigid objects for which the assumption of purely linear vibration dynamics made in §2.2.1 is invalid. We propose a procedural method for synthesizing realistic sounds due to nonlinear thin-shell vibrations. We use the linear modal analysis technique presented in §2.2.1 to generate a small-deformation displacement basis, then couple the modes together using nonlinear thin-shell forces. To enable audio-rate time-stepping of nonlinear modal vibrations with mesh-independent cost, we propose a reduced-order dynamics model based on a thin-shell cubature scheme. Limitations such as mode locking and pitch glide are addressed. To support fast evaluation of mid-frequency mode-based sound radiation for detailed meshes, we propose *far-field acoustic transfer maps* (FFAT maps) which can be pre-computed using state-of-the-art fast Helmholtz multipole methods. Familiar examples are presented including rumbling trash cans and plastic bottles, crashing cymbals, and noisy sheet metal objects, each with increased richness over linear modal sound models.

3.1 Introduction

The linear modal sound pipeline discussed in §2.2.3 is widely used for rigid bodies in computer animation and virtual environments [129, 100, 12]. When combined with acoustic transfer models for sound radiation [70] they can provide convincing physically based sound sources, especially for pure ringing tones such as chimes, bells, or "knocks." Unfortunately, we lack effective sound models for a broad class of noisy virtual objects: thin shells (objects with thicknesses orders of magnitude smaller than their other dimensions). Thin shells are very common in real and virtual environments (sheet metal objects (trash cans, oil drums, tin roofs, machinery), plastic containers (water

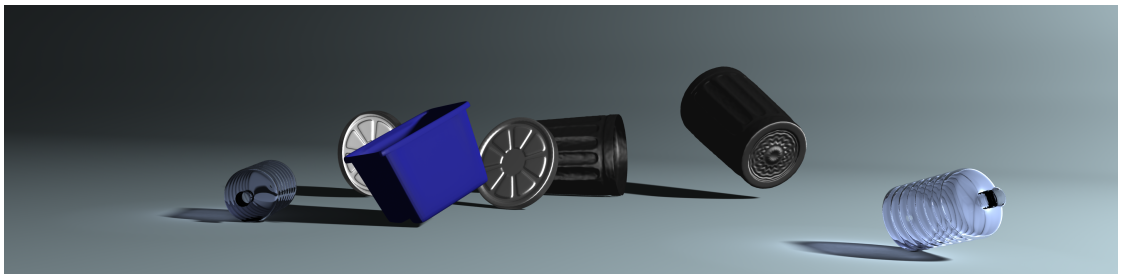


Figure 3.1: **Crash!** Our physically based sound renderings of thin shells produce characteristic "crashing" and "rumbling" sounds when animated using rigid body dynamics. We synthesize nonlinear modal vibrations using an efficient reduced-order dynamics model that captures important nonlinear mode coupling. High-resolution sound field approximations are generated using Far-Field Acoustic Transfer (FFAT) maps, which are precomputed using efficient fast Helmholtz multipole methods, and provide cheap evaluation of detailed low- to high-frequency acoustic transfer functions for realistic sound rendering.

bottles), musical instruments (cymbals), etc.), and produce rich and easily recognizable impact sounds. Their rich nonlinear vibrations produce distinct "crashes" and "rumbles" that are poorly approximated by linear modal sound models which lack nonlinear mode coupling. To make matters worse, thin shells are often very loud and important sound sources due to their ability to vibrate and radiate sound so effectively, e.g., consider a metal roof pelted by hail. Alas, their expensive nonlinear dynamics have made thin shells computationally impractical for physically based sound synthesis.

In this chapter, we propose an efficient method for synthesizing realistic sounds from thin-shell structures undergoing small but nonlinear vibrations. Given a description of an object's geometry and material properties, we compute linear vibration modes and couple these modes together using a nonlinear thin-shell force model. To accelerate nonlinear modal dynamics, we optimize a thin-shell cubature scheme to evaluate reduced-order shell forces at costs independent of the geometric complexity of the model. We show that the complex internal dynamics of thin-shell models can be approximated with sufficient accuracy and efficiency to allow practical synthesis of plausible thin-shell sounds. We also address sound-related locking effects that arise when sim-

ulating nonlinear modal dynamics that might produce pitch-glide artifacts in general animations. Our nonlinear reduced-order dynamics model can synthesize modal vibrations using hundreds of modes, which then drive sound radiation. Unfortunately the estimation of sound wave radiation via prior acoustic transfer models is complicated for two reasons: (1) the nonlinear mode vibrations are no longer linear harmonics, and (2) higher frequency acoustic transfer with high-resolution meshes is expensive to precompute, represent, and evaluate at runtime. First, we observe that nonlinear thin-shell vibrations produced by our animations exhibit frequency-localized modes for which linear frequency-domain radiation models still provide a plausible approximation. Second, we propose **far-field acoustic transfer maps** (FFAT maps) for fast runtime evaluation of high-frequency acoustic transfer from general modal vibration sources. Our texture-based approach leverages state-of-the-art fast Helmholtz multipole methods to precompute acoustic transfer functions for complex thin-shell (or more general) structures, while delivering the simplicity and speed of texture sampling for runtime transfer evaluation. At runtime, sounds are synthesized by time-stepping the nonlinear reduced-order model to estimate modal amplitudes, which are then multiplied by acoustic transfer values to auralize the thin-shell sound source (see, e.g., (2.24)).

Other Related Work: Deformable thin shell models have seen widespread use in computer animation, especially for large-deformation simulations of parameterized cloth models [125, 7]. Plate-like cloth models with flat rest configurations are most common. However shell models for objects with intrinsically curved rest configurations have recently gained attention, e.g., to model clothing with folds and wrinkles [13]. While various thin-shell models exist in the mechanics literature (see [25]), discrete shell models that better meet computer animation needs have appeared recently. These models use simple bending energy formulations that allow standard cloth solvers to produce convincing large-deformation thin-shell dynamics [57, 13]. The results presented in

this chapter use the elastic thin-shell model summarized in Gingold et al. [54]. Other works outside of the computer graphics community have aimed to make shell simulations faster and more physically realistic [29, 28, 56]. For example, Green et al. [56] propose a multigrid-based method for subdivision shells to accelerate the solution of linear systems involved in thin-shell dynamics. The mathematical structure of bending energy formulations has been exploited and simplified to accelerate near-isometric deformation of thin plates [9] and shells [52]. Our work differs in that we do not require large-deformation animations or sophisticated linear system solvers, but rather focus on small mode-related shell vibrations and try to obtain explicit time-stepping costs sublinear in geometric complexity to enable audio-rate sound synthesis. We also prefer mode-based representations since they are preferred for auralization with frequency-domain acoustic transfer models.

Linear modal vibration models are well known in animation [102, 71]. They have proven effective for procedural sound synthesis, and are increasingly used to simulate rigid-body impact sounds [2, 31, 128, 129, 100] in part due to excellent synthesis speed for interactive applications [129, 104, 12]. "Modal warping" has been used to approximate large-deformation thin shells for animation [26], but such models are of limited use for nonlinear sound synthesis since the underlying linear modal oscillators are uncoupled by design. There is a tremendous amount of work on the nonlinear vibrations of plates and shells in engineering [96, 94] and related structure borne sound [32]. Important examples are nonlinear vibrations in musical instruments [44], such as gongs and cymbals [22]. Simple all-pass nonlinear passive filters have been used to mimic nonlinear mode-coupling effects [103]. For nonlinear modal analysis, linear eigenmodes are often used for nonlinear subspace integration of dynamics to resolve weak material nonlinearities in small-strain configurations and resolve mode-mode coupling [8] (for a good discussion on nonlinear mode coupling in beams and plates see [86]). Nonlinear

normal modes [97, 127] have been used to describe a nonlinear vibration mode's shape as a linear superposition of other linear modes, i.e., to resolve nonlinear mode coupling. However, likely due to the high computational complexity of simulating nonlinearly coupled modal models (often $O(r^4)$ or worse), we are unaware of sound synthesis results in the literature that demonstrate results comparable to ours, i.e., with several hundred fully coupled modes. We achieve this by extending cubature optimization techniques of An et al. [3]; their method estimates volumetric cubature schemes, and was even used to evaluate nonlinear modal shell vibrations for sound synthesis, but the thin shell had to be modeled using several hundred thousand tetrahedral elements. We extend cubature schemes to handle thin shells more efficiently, and obtain cubature-based reduced-order modal models with $O(r^2)$ time-step complexity for r nonlinearly coupled modes.

The only physically based sound rendering work in graphics that addresses nonlinear object vibrations is O'Brien et al. [99]. They use an explicitly integrated large-deformation finite element model to simulate nonlinear vibrations of objects using small time-step sizes, and a time-domain ray-based "Rayleigh method" (a.k.a. direct propagation) to approximate sound radiation. Interesting results were obtained for short animations with large deformations and buckling. Unfortunately simulation times were on the order of a day (circa 2001) for models of rather modest geometric complexity ($<$2000 tetrahedra), and the expensive radiation model is known to have limited accuracy [34]. Bilbao has considered energy conserving finite difference discretizations and time-stepping schemes for nonlinear plates to generate plausible sounds [10]. In contrast to these works, we focus on efficient sound models for nonlinearly forced thin-shell vibrations: we develop efficient subspace integration techniques for geometrically complex nonlinear modal models which drive frequency-domain acoustic transfer models that are consistent with frequency-domain wave radiation.

The most closely related work on sound radiation to ours is "Precomputed Acoustic Transfer" [70], which augmented a linear modal sound model with a multipole-based Helmholtz approximation of each mode's acoustic transfer function, $p(\boldsymbol{x})$. The approach enables real-time computation of vibration and sound radiation for geometrically complex objects. Unfortunately due to the increasing complexity of mid- to high-frequency radiation (higher kL values) [34], increasingly complex multipole approximations are required at higher frequencies; such models are costly to precompute, costly to evaluate at runtime, and become increasingly inaccurate at difficult higher frequencies. There is significant work in the acoustics community on reconstructing acoustic quantities (such as sound source multipole expansion coefficients) from often sparse acoustic pressure measurements, e.g., for near-field acoustic holography, but most methods are limited to low- to mid-frequency problems [131, 134]. Our far-field acoustic transfer (FFAT) maps provide a simple method for constant-time transfer evaluation which handles low- to high-frequency far-field radiation complexity by exploiting (1) fast multipole boundary element methods from acoustics [83, 119], and (2) texture-based far-field expansions that are well suited to capturing rapid angular variations with simple radial structure.

Finally, it is possible in principle to simulate fully nonlinear vibrations coupled to 3D (nonlinear) sound fields. For instance, two-dimensional simulations of a nonlinear plate coupled to acoustic fluid have been modeled using the nonlinear Euler equations and used to produce nonlinear far-field radiation [49]. Unfortunately such approaches are currently computationally unappealing for 3D audio-rate simulations of nonlinear sound.

3.2 Nonlinear Modal Vibrations for Thin Shells

3.2.1 Background: Thin-Shell Dynamics

We consider discrete thin shells modeled using triangle meshes with N_Δ triangles, and N_v vertices. Following a suitable discretization via the finite element (or other) method we obtain an N-dimensional system of ordinary differential equations (recall, also, (2.10)),

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{D}\dot{\mathbf{u}} + \mathbf{f}_{int}(\mathbf{u}) = \mathbf{f}_{ext} \tag{3.1}$$

where $\mathbf{u} = \mathbf{u}(t) \in \mathbb{R}^N$ are mesh vertex displacements, $\dot{\mathbf{u}}$ are velocities, $\ddot{\mathbf{u}}$ are accelerations, $\mathbf{M} \in \mathbb{R}^{N\times N}$ is the mass matrix; $\mathbf{f}_{int}(\mathbf{u})$ describes nonlinear internal thin-shell forces; and $\mathbf{f}_{ext}$ are time-dependent external forces such as gravity or contact forces. For lightly damped small vibrations, we use linear Rayleigh damping [8] with $\mathbf{D} = \alpha\mathbf{M} + \beta\mathbf{K}$, with $\mathbf{K}$ the stiffness matrix (Jacobian of $\mathbf{f}_{int}$) evaluated at $\mathbf{u} = 0$.

Without loss of generality, we use the elastic shell model of [54], in part because it is based on physical material parameters that simplify parameter tuning for sound synthesis. The deformation strain energy, $E(\mathbf{u})$, is an integral over the surface of the strain energy density, $W(\mathbf{u}; X)$, where X is a material position on the undeformed surface. The strain energy density is decomposed in to two parts

$$W = W_m + W_b \tag{3.2}$$

where W_m is the membrane strain energy density which penalizes tangential stretching or compression; and W_b is the bending energy density which resists bending away from the rest configuration. The membrane and bending strain energy densities are defined in

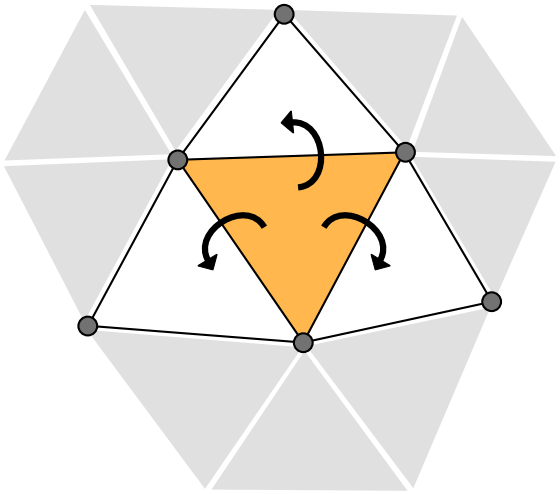


Figure 3.2: **Stencil of triangle element:** Whereas the piecewise constant membrane strain energy density W_m can be determined for a given triangle by evaluating only that triangle's vertex positions, determination of the triangle's bending strain energy density also requires the state of neighboring triangles. As a result, evaluating the full strain energy density for this triangle requires that we consider the positions of the 6 vertices highlighted here.

terms of per-triangle strain tensors:

$$W_m = \frac{Y\,h}{2(1-\nu^2)}\,\left[(1-\nu)\,\mathrm{tr}(\boldsymbol{\epsilon}_m^2) + \nu\,\mathrm{tr}(\boldsymbol{\epsilon}_m)^2)\right] \tag{3.3}$$

$$W_b = \frac{Y\,h^3}{24(1-\nu^2)}\,\left[(1-\nu)\,\mathrm{tr}(\boldsymbol{\epsilon}_b^2) + \nu\,\mathrm{tr}(\boldsymbol{\epsilon}_b)^2)\right] \tag{3.4}$$

where h is the shell thickness, Y is Young's modulus, and ν is Poisson's ratio (see Gingold et al. [54] equations for $W_{membrane} = W_m$ and $W_{bending}^{Koiter} = W_b$). The membrane and bending strains are $\boldsymbol{\epsilon}_m$ and $\boldsymbol{\epsilon}_b$, respectively; $\boldsymbol{\epsilon}_m$ is a 3×3 tensor which varies as triangle edges deviate from their rest lengths; and $\boldsymbol{\epsilon}_b$ is a 3×3 tensor which changes as the dihedral angles between a triangle and its three neighbors vary from the corresponding dihedral angles in the shell's rest pose (see [54] for definitions). Consequently, the strains and strain energy density, W, can be defined as piecewise constant over shell triangles, with each triangle's W_m value a function of the triangle's 3 vertex positions, whereas its bending strain energy, W_b, is a function of 6 vertices (see Figure 3.2).

It follows that the deformation energy is given by the integral over the undeformed surface, S:

$$E(\mathbf{u}) = \int_S W(\mathbf{u}; X)\, dS_X = \sum_{i=1}^{N_\Delta} A_i W_i(\mathbf{u}) \tag{3.5}$$

where W_i is the piecewise constant strain energy density for triangle i, and A_i is its area. The desired internal thin-shell force is

$$\mathbf{f}_{int}(\mathbf{u}) = \nabla_{\mathbf{u}} E(\mathbf{u}) = \sum_{i=1}^{N_\Delta} A_i\, \nabla_{\mathbf{u}} W_i(\mathbf{u}). \tag{3.6}$$

which gathers both membrane and bending contributions.

3.2.2 Reduced-order Thin-Shell Dynamics

We now describe the nonlinear mode-coupled dynamics model used for sound synthesis.

Linear Modal Analysis: As a first step, we compute r linear eigenmodes using the mass matrix $\mathbf{M}$ and rest-pose stiffness matrix $\mathbf{K}$ of the thin-shell system, given by

$$\mathbf{K} = \left.\frac{\partial \mathbf{f}_{int}}{\partial \mathbf{u}}\right|_{\mathbf{u}=\mathbf{0}}. \tag{3.7}$$

This is analysis uses standard methods [118, 8] which are reviewed in §2.2.1. Following the notation of §2.2.1, this procedure yields a basis of eigenmodes $\mathbf{U} \in \mathbb{R}^{N \times r}$ and associated angular frequencies ω_i, along with the diagonal matrix $\mathbf{\Lambda} = \text{diag}(\omega_1^2, \ldots, \omega_r^2)$.

Nonlinear Subspace Integration: To obtain nonlinear coupling between linear modes we employ dimensional model reduction [8, 76] by substituting $\mathbf{u} = \mathbf{U}\mathbf{q}$ in to the full-dimensional equations of motion (3.1), and premultiplying by $\mathbf{U}^T$ to project in to the r-dimensional modal subspace:

$$\ddot{\mathbf{q}} + \widetilde{\mathbf{D}}\dot{\mathbf{q}} + \widetilde{\mathbf{f}}_{int}(\mathbf{q}) = \widetilde{\mathbf{f}}_{ext} \tag{3.8}$$

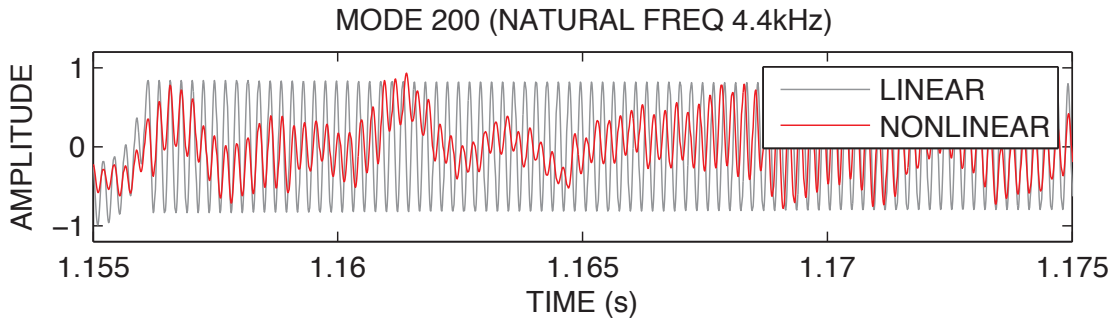


Figure 3.3: **Nonlinear mode coupling:** The nonlinear and linear modal dynamics of $q_{200}(t)$ are compared for the ride cymbal's response to a single metal ball impact (the first video example). The nonlinear mode exhibits rich dynamics, and strong coupling to lower frequency modes.

where

$$\widetilde{\mathbf{D}} = \alpha\mathbf{I} + \beta\mathbf{\Lambda} \tag{3.9}$$

$$\widetilde{\mathbf{f}}_{int}(\mathbf{q}) = \mathbf{U}^T\mathbf{f}_{int}(\mathbf{U}\mathbf{q}) \tag{3.10}$$

$$\widetilde{\mathbf{f}}_{ext} = \mathbf{U}^T\mathbf{f}_{ext}. \tag{3.11}$$

The linear dynamics formulation presented in §2.2.1 considers forces of the form $\widetilde{\mathbf{f}}_{int} = \mathbf{\Lambda}\mathbf{q}$, which require only $O(r)$ time to evaluate since $\mathbf{\Lambda}$ is diagonal. Meanwhile, (3.8) provides nonlinear mode coupling when integrated (see Figure 3.3). Unfortunately, explicit time-stepping of these nonlinear equations at audio rates (44.1 kHz; $\Delta t \approx 2.3 \times 10^{-5}$) is quite expensive due to $\widetilde{\mathbf{f}}_{int}$ evaluation, which involves subspace-projection of $\mathbf{f}_{int}$ in (3.6) via a gather over N_Δ triangles – an undesirable $O(r\,N_\Delta)$ cost per audio timestep.

3.2.3 Thin-Shell Cubature Scheme

We accelerate the evaluation of internal forces $\widetilde{\mathbf{f}}_{int}$ by extending the volumetric approach of An et al. [3] to optimize cubature schemes for thin shells. We use a compound cubature scheme that evaluates both bending and stretching force integrals simultaneously

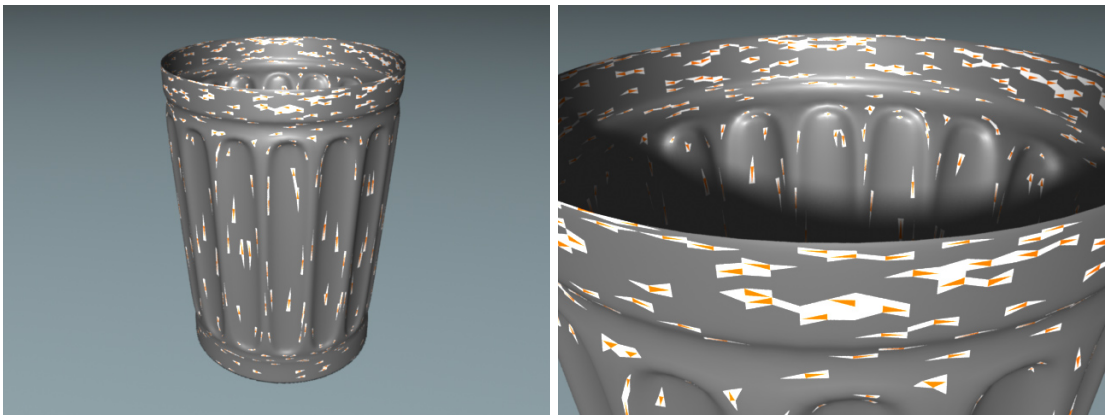


Figure 3.4: **Illustration of cubature scheme:** (Left) Trash can (200 modes) with 800-feature cubature scheme; (Right) close-up of triangle-flap features.

thereby sharing vertex-deformation computations. We make the approximation

$$\widetilde{\mathbf{f}}_{int}(\mathbf{q}) = \mathbf{U}^T \mathbf{f}_{int}(\mathbf{U}\mathbf{q}) = \sum_{i=1}^{N_\Delta} A_i \, \mathbf{g}_i(\mathbf{q}) \tag{3.12}$$

$$\approx \sum_{i \in \mathcal{C}} w_i \, \mathbf{g}_i(\mathbf{q}), \tag{3.13}$$

where $\mathbf{g}_i(\mathbf{q}) \equiv \mathbf{U}^T \nabla_{\mathbf{u}} W_i(\mathbf{U}\mathbf{q})$, and w_i are cubature weights, and $\mathcal{C}$ is a set of integers defining triangles in which cubature points are chosen. See Figure 3.4 for a visualization of this triangle set for one of our example models.

Cubature Training: Since the strain energy density (and thus force density) defined in the shell model is piecewise-constant over a triangle element, it suffices to only consider one cubature point per triangle during the optimization pre-process. However, at run-time the triangle and its three edge flaps must be reconstructed for each cubature point, as the bending force is dependent on the state of neighboring triangles. We also considered alternate cubature schemes, including optimizing separate weights for the two force integrals, but this did not significantly affect convergence behavior. To generate training samples for cubature optimization, we randomly sample a Gaussian for each mode to get physically plausible training poses [3]. Convergence plots are shown in Figure 3.5.

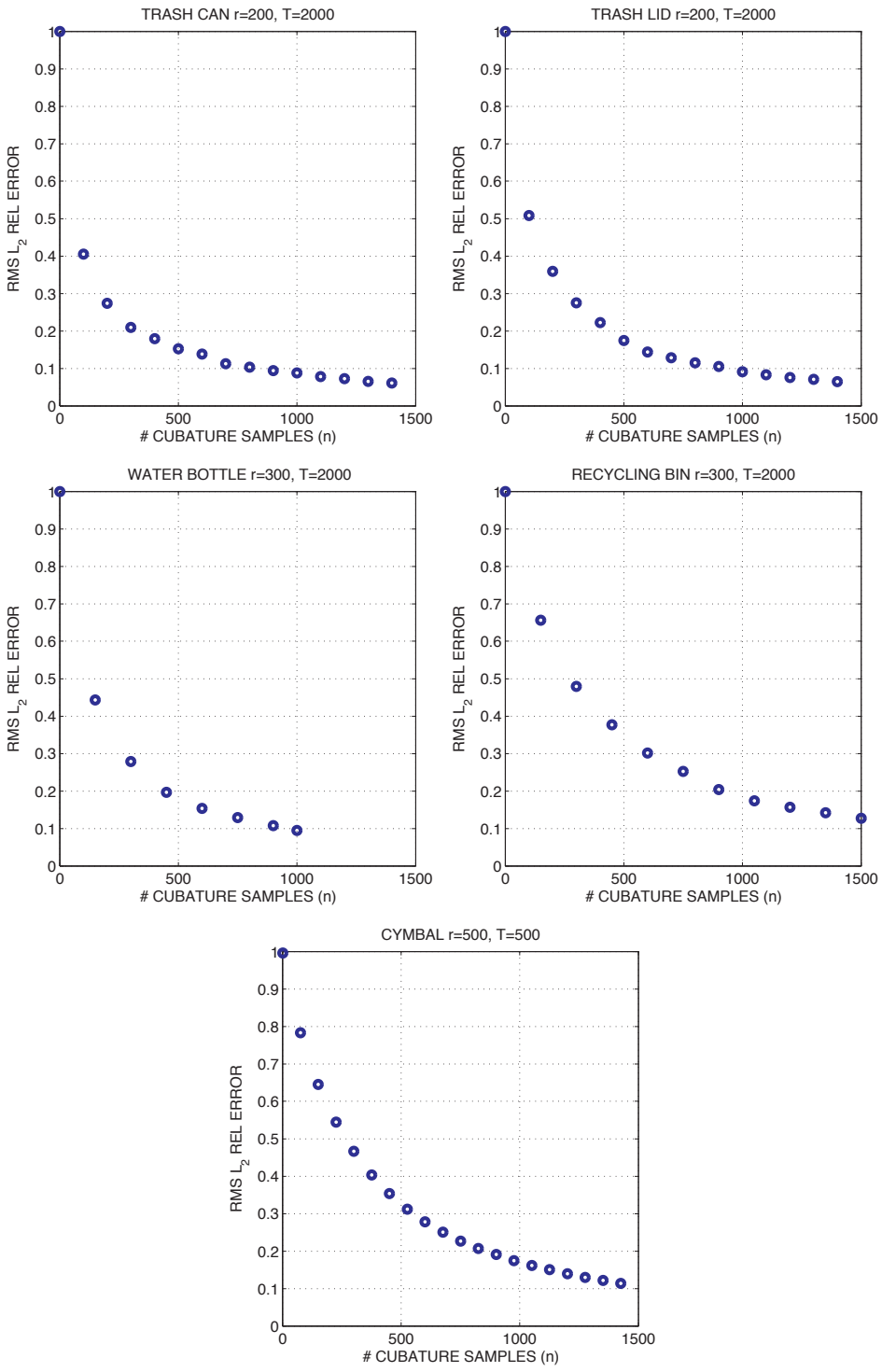


Figure 3.5: **Cubature training convergence plots** reveal that $\sim 10\%$ error is often obtained after $n = 4r$ cubature features, which is higher than the volumetric modal models in An et al. [3].

$O(r^2)$ **Force Evaluation:** In practice, we can construct cubature schemes to modest accuracies (e.g., 10% relative error) for which the number of cubature samples $|\mathcal{C}| = O(r) \ll N_\Delta$. The resulting thin-shell cubature scheme can approximate $\widetilde{\mathbf{f}}_{int}(\mathbf{q})$ at $O(r^2)$ cost. More important than complexity is that the scheme is fast in practice. Achieving full nonlinear coupling of r modes at $O(r^2)$ cost makes offline sound synthesis practical for several hundred modes (see Table 3.2). To understand this cost, note that each $\mathbf{g}_i(\mathbf{q})$ r-vector in (3.13) can be evaluated in $O(r)$ flops using three steps: given the triangle element's 6 stencil vertices, V (see Figure 3.2), we (1) evaluate the 6 vertex displacements, $\mathbf{u}_V = \mathbf{U}_V\mathbf{q}$ in $O(r)$ flops; (2) evaluate the 6 vertex forces $\mathbf{f}_V = \nabla_{\mathbf{u}_V} W_i(\mathbf{u}_V)$ in $O(1)$ flops, then (3) project the vertex forces in to the r-dimensional subspace, $\mathbf{g}_i = \mathbf{U}_V^T\mathbf{f}_V$, in $O(r)$ flops.

3.3 Limiting Artificial "Pitch Glide"

Rigid body animations can generate a wide range of contact impulses, including violent impacts that would dent or damage the shell in reality. Unlike the linear modal sound model which simply gets proportionally louder with increased impulse strength, we need to take precautions to ensure that the nonlinear modal model operates in a valid energetic range to avoid "mode locking" [8]. Intuitively speaking, hard forcing can produce mode locking when the model has insufficient degrees of freedom to deform due the limitations of the linear modal basis. Without energetic limits, listeners may perceive a nonphysical "pitch glide" artifact during contact, during which vibration frequencies start out very high, but then glide down to their natural frequencies (see Figure 3.6). While "pitch glide" occurs naturally for some objects, e.g., Chinese opera gongs [44], locking in low-frequency modes can also artificially increase the effective numerical stiffness of coupled high-frequency modes, e.g., by stretching the shell, which

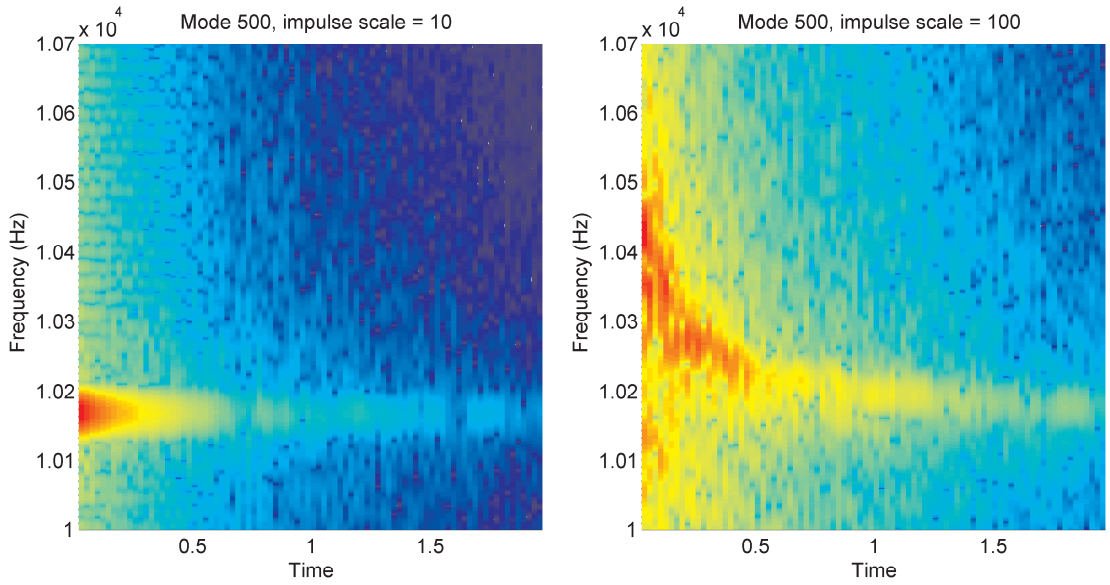


Figure 3.6: **Illustration of locking-related "pitch glide:"** Spectrograms are shown for virtual cymbal sounds resulting from different impulse magnitudes. As the impulse magnitude grows, one can see an increasingly noticeable frequency drop at the beginning of the spectrograms.

introduces pitch-glide artifacts (also see Figure 3.8). Numerically we would also like to avoid locking since that nonlinearity can introduce severe time-step restrictions; in our implementation, we use a fixed-rate timestep to avoid temporal artifacts, so smaller timesteps are particularly undesirable. In a full degree-of-freedom model, the elements would be free to bend yet still avoid excess stretching, so the frequencies remain roughly constant. Unfortunately, full simulations are often impractically expensive and numerically unstable compared to reduced simulation, so we propose a simple technique for limiting pitch-glide artifacts in practice.

Impulse Limiter: To avoid forcing the nonlinear oscillator to high energies, we use a simple per-timestep impulse filter that bounds the system's vibrational energy. Given the desired velocity impulse, $\Delta\dot{\mathbf{q}} = \Delta t\, \widetilde{\mathbf{f}}_{ext}$, we introduce an (as yet unknown) scale factor, $0 \leq \alpha \leq 1$, and only apply the scaled impulse, $\alpha\Delta\dot{\mathbf{q}}$. We estimate the total energy

Algorithm 1: Compute α for impulse limiter

```
begin
    if E'(1) ≤ E_max or E'(1) < E then
        return α = 1 ;
    else if E > E_max then
        return α = 0 ;
    else
        solve E'(α*) = E_max for α* ∈ [0, 1] ;
        return α* ;
end
```

(kinetic + potential) before the impulse as,

$$\mathcal{E} = \dot{\mathbf{q}}^T \dot{\mathbf{q}} + \mathbf{q}^T \mathbf{\Lambda} \mathbf{q}, \tag{3.14}$$

and the α-parameterized post-impulse energy as,

$$\mathcal{E}'(\alpha) = (\dot{\mathbf{q}} + \alpha \Delta\dot{\mathbf{q}})^T (\dot{\mathbf{q}} + \alpha \Delta\dot{\mathbf{q}}) + \mathbf{q}^T \mathbf{\Lambda} \mathbf{q}. \tag{3.15}$$

We set a maximum allowed energy, $\mathcal{E}_{max} = \sigma M$ by specifying σ, the maximum energy per unit mass, where M is the object's total mass; in our implementation, we use the same σ value for all scene objects (typically $\sigma \approx 1 - 4$). We determine the impulse limiter's α value using Algorithm 1.

3.4 Mapping Far-Field Acoustic Transfer

We now describe a general method for approximating acoustic transfer using far-field acoustic transfer (FFAT) maps. See §2.1.2 (and [70]) for background on acoustic-transfer-based sound rendering of modal models. Our approach involves three steps: (1) for each mode we precompute detailed pressure samples on concentric exterior spherical surfaces using commodity Helmholtz boundary integral solvers, then (2) we precompute a low-order Laurent expansion for each outgoing ray direction; then (3) at runtime we

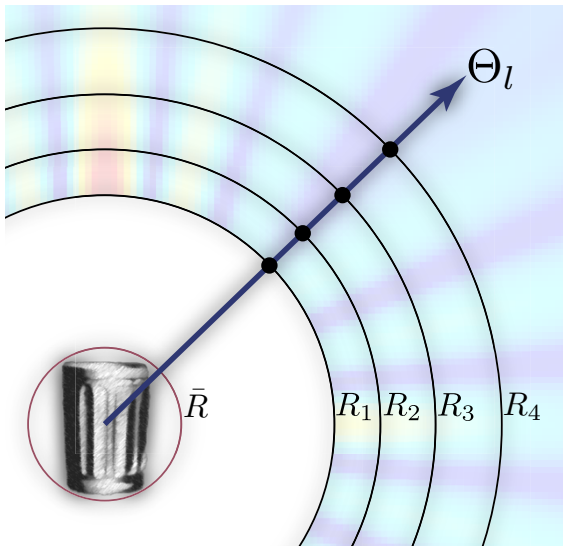


Figure 3.7: **Geometry of FFAT Map Estimation**

can evaluate a low-order expansion of any mode's transfer value at a far-field listening position using $O(1)$ operations. This data-driven approach avoids the use of multi-point multipole expansions (as in James et al. [70]) which can be complex to fit and evaluate for higher frequency radiation [134].

Far-Field Acoustic Transfer (FFAT) Maps: To accelerate render-time acoustic transfer evaluation, we propose a simple approximation to the pressure field, $p(\boldsymbol{x})$, motivated by the far-field, asymptotic M-term series expansion [59]

$$p(\boldsymbol{x}) \sim h_0(kR) \sum_{j=1}^{M} \frac{\Psi_j(\theta, \phi)}{(kR)^{j-1}} \tag{3.16}$$

for $\boldsymbol{x} = spherical(\theta, \phi, R)$, where h_0 is the monopole-like 0^{th}-order spherical Hankel function of the first kind, $h_0(kR) = -\frac{ie^{-ikR}}{kR}$, and the Ψ_j functions describe the angular dependence of the pressure field. In practice we seek an M-term expansion where M is very small, e.g., $M = 1 \ldots 4$.

We produce least-squares estimates of Ψ_j for a given M as follows. Given a coordinate system defined at the center of the object (we use the center of mass), we define

a fixed set of angular directions, $\Theta_l = (\theta_l, \phi_l)$, and a set of radii $R_1, \ldots, R_K$, $K > M$ (see Figure 3.7). Using a Helmholtz boundary integral solver, we rasterize a reference $p(\boldsymbol{x})$ solution at all (R_i, Θ_l) locations; we use the ***FastBEM Acoustics*** implementation (`www.fastbem.com`) of the fast multipole boundary element method [83, 119]. In our examples, we use $K = 6$ shells to estimate between 1 and 4 Ψ_j maps, rasterizing $\theta \in [0, \pi]$ in to T values, and $\phi \in [0, 2\pi]$ in to $2T$ values where T is a function of wave number. Given a radius $\bar{R}$ for an object's bounding sphere about its center of mass, we choose radii $R_1, \ldots, R_K$ in a geometric sequence $R_i = 2.5\bar{R}\gamma^{i-1}$ (we used $\gamma = 1.54$). Using this pressure data, each angular direction, Θ_l, has the following complex-valued, $K \times M$ least-squares problem,

$$\sum_{j=1}^{M} \frac{h_0(kR_i)}{(kR_i)^{j-1}} \Psi_j(\Theta_l) = p(R_i, \Theta_l) \quad \Leftrightarrow \quad \sum_{j=1}^{M} A_{ij}\Psi_{jl} = p_{il}, \quad i = 1, \ldots, K. \quad (3.17)$$

We abbreviate (3.17) as $\mathbf{A}\boldsymbol{\Psi} = \mathbf{P}$. The system matrix $\mathbf{A}$ is the same for each angular direction; therefore, we can solve all directions simultaneously. We also note that different rows of $\mathbf{P}$ have different radii, and can differ greatly in magnitude. To address this, we use a weighted least squares approach that normalizes by the RMS magnitude of each $\mathbf{P}$ row. Specifically, let $\mathbf{W}$ be a diagonal weighting matrix with $\mathbf{W}_{ii} = 1/\|\mathbf{P}_{i,:}\|$, then we solve the weighted least-squares problem, $\mathbf{WA}\boldsymbol{\Psi} = \mathbf{WP}$ using TSVD to obtain $\boldsymbol{\Psi} = (\mathbf{WA})^{\dagger}(\mathbf{WP})$ ($\mathbf{P}_{i,:}$ is shorthand for the i^{th} row of $\mathbf{P}$). Once $\boldsymbol{\Psi}$ is evaluated, each row of this matrix stores the values of a series term Ψ_j evaluated in all discrete directions. We can extract these rows and store them as floating-point textures for subsequent evaluation.

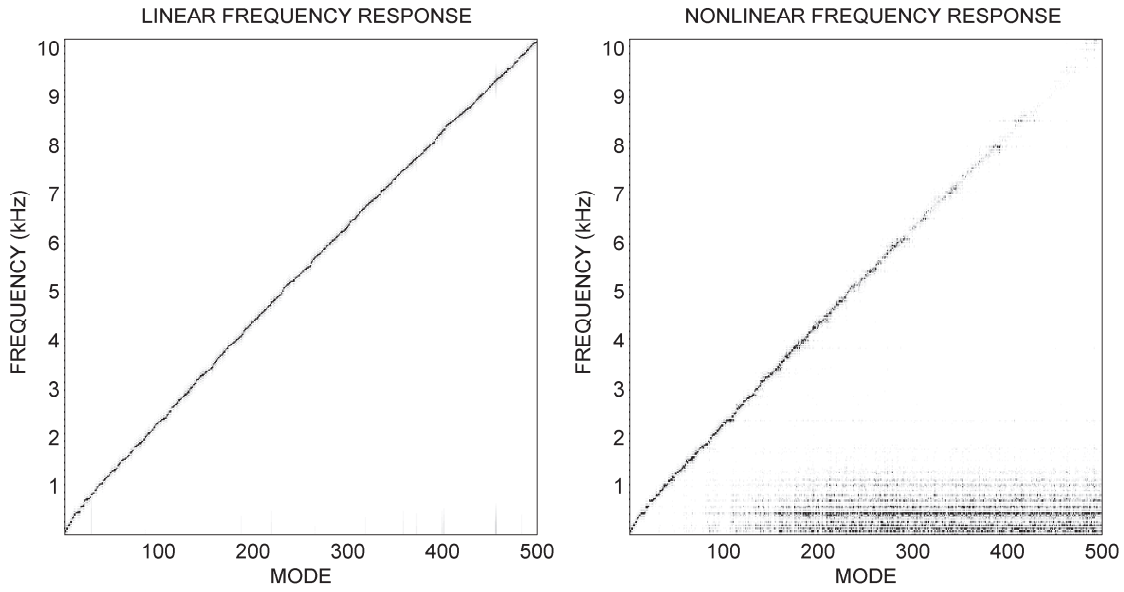
Varying FFAT map resolution: In our implementation, all FFAT maps are computed via a uniform sampling of angle (θ-ϕ) space. That is, given some number of θ divisions T, we store FFAT map terms and precompute samples on each spherical shell at $2T^2 + 2$ angular positions. In general, the acoustic transfer function's angular com-

Model	L (m)	tri	vtx	N	modes	freq (kHz)	material	ν	Y (GPa)	h (mm)	α	β (10^{-9})	n_{cuba}	Error$_{cuba}$	kL	Δt (s)
Trash Can	0.75	77536	38833	116499	200	0.071 – 4.43	Steel	0.30	190	2	0.5	75	800	10.3%	0.98 – 61	1/44100
Trash Lid	0.55	34312	17286	51858	200	0.112 – 6.79	Steel	0.30	190	2	0.5	75	800	11.5%	1.1 – 68	1/44100
Water Bottle	0.46	28658	14418	43254	300	0.116 – 3.59	Polycarb.	0.37	2.4	2.25	0.5	400	900	10.7%	0.98 – 48	1/44100
Recycling Bin	0.61	109568	54945	164835	300	0.062 – 2.21	Polycarb.	0.37	2.4	5	4.0	300	1200	15.7%	0.70 – 30	1/44100
Cymbal	0.50	61952	31104	93312	500	0.061 – 9.94	Bronze	0.33	124	0.7	1.0	6.25	1500	10.7%	0.57 – 92	1/88200

Table 3.1: **Model Statistics:** L refers to the length of the object along its longest axis. *tri* and *vtx* refer to the number of triangles and vertices used to discretize the object. *modes* refers to the number of vibration modes used for sound synthesis and the *freq* column provides the frequency range for these modes. ν, Y and h are the Poisson ratio, Young's modulus and thickness of the thin shell material, respectively. α and β are Rayleigh damping parameters. n_{cuba} is the number of cubature points used for force evaluation, and Error$_{cuba}$ is the relative residual error for forces evaluated using this scheme on the poses used for cubature training.

plexity increases with modal frequency (see figures 3.12 and 3.13) suggesting that some modes require greater angular FFAT map resolution than others. We use a simple model in which T varies linearly with the wave number k: $T = \lceil ck + d \rceil$. In our experiments, we found that a base resolution of $d = 15$ and a frequency-dependent parameter of $c = 2.25m$ was sufficient to adequately capture the angular complexity of acoustic transfer functions.

Discussion of frequency localization: We have assumed that the nonlinear modal vibrations are approximately time-harmonic with frequencies similar to the linear modal vibrations. While this is true for weak forcing, it is far less true for hard forcing (see Figure 3.8). Nevertheless, we believe that linear frequency-domain acoustic transfer provides a cheap yet plausible sound model. The alternative evaluation of 3D time-domain wave radiation is significantly more expensive (and less appealing) than the $O(r)$/object runtime evaluation of FFAT map transfer.



Linear Response Nonlinear Response

Figure 3.8: **Frequency spectrum of $\mathbf{q}(t)$ for a hard cymbal "crash"** (first example in video): The linear model (Left) illustrates that each modal coordinate $q_i(t)$ is strongly localized in the frequency domain about its modal frequency, ω_i, whereas the nonlinear modal model (Right) exhibits a more complex response that is frequency-localized for lower-frequency modes, but higher modes become increasingly coupled to low-frequency modes—a possible sign of modal locking after this strong forcing.

3.5 Results

We now describe numerical and sound experiments for several models and multibody collision scenarios. Please see the result video for the paper associated with this work [18] for all animation and sound rendering results [1]. Model statistics are provided in Table 3.1. Representative timings are given in Table 3.2.

We provide sound comparisons between four cases:

1. **Nonlinear with transfer ("Harmonic Shells"):** Nonlinear modal vibrations with acoustic transfer (FFAT maps, or fast multipole method evaluation).

2. **Linear with transfer:** Linear modal vibrations with acoustic transfer. This case

[1] `http://www.cs.cornell.edu/projects/harmonicshells/`

Model	Modes r	Modal Analysis	Cubature Precomp.	Timestep Cost	Simulation Cost (per second of audio)	FFAT Precomp. (average time/mode)	FFAT Eval (all modes, $M=4$)	FFAT Storage (floats, $M=1$)
Trash Can	200	569 s	2.49 hr	16.1 ms	714 s	109.2 min	0.151 ms	56 MB
Trash Lid	200	170 s	1.87 hr	14.6 ms	642 s	85.5 min	0.151 ms	113 MB
Water Bottle	300	314 s	4.31 hr	23.6 ms	1026 s	25.6 min	0.227 ms	54 MB
Recycling Bin	300	2332 s	9.65 hr	27.8 ms	1224 s	48.0 min	0.227 ms	25 MB
Cymbal	500	1155 s	3.88 hr	44.3 ms	3900 s	318 min	0.378 ms	512 MB

Table 3.2: **Representative Timings:** All timings are for a single 2.66GHz Xeon X5355 processor core, except "Cubature Precomp" which used 8 cores.

typically sounds plausible, but misses characteristic nonlinear "crash" and "rumble" effects, and timbre variations with amplitude.

3. **Linear with monopole: transfer** Linear modal vibrations with the low-frequency, far-field monopole radiation model (equation (15) in [70]; also used in [12]). Lacking both nonlinear vibrations and acoustic transfer, this case usually sounds quite unrealistic.

4. **Nonlinear with monopole: transfer** For comparison, we also render nonlinear modal vibrations with the far-field monopole model. While the vibrations are nonlinear, without acoustic transfer the sound quality is poor.

Implementation Details: We precompute dominant linear vibration modes using MATLAB's [88] generalized eigenvalue solver (using the shift-and-invert spectral transformation implemented in ARPACK [80]). To improve our graphics model's mesh quality for vibration and radiation analysis, we remesh the shells (using GNU GTS). We exploit mode-level parallelism to precompute acoustic transfer models (fast multipole solves, and FFAT map estimation) on a 16-node cluster (8-core, 2.66GHz, 8GB, Xeon X5355 processor nodes). All animations are performed using rigid body dynamics with vibration models defined in the appropriate rigid body frame [117]. Collisions are detected using a rigid sphere-tree bounding volume hierarchy, and resolved using a linear Kelvin-Voigt penalty contact model. Rigid body dynamics are time-stepped using symplectic Euler at rates sufficient to resolve penalty contact forces; modal vibrations are

time-stepped (explicit subspace Newmark) at audio rates (e.g., 44100Hz); and acoustic transfer is evaluated at 1000Hz along the two-ear listening trajectory. A simple contact damping model is used to damp vibrations of objects in ground contact. In our simulation pipeline, we first simulate rigid-body motion and dump subspace force impulses to disk, then in a second pass we compute modal vibrations and synthesize sound at the listening position. Each object's final sound is computed as a linear superposition of modal contributions with a simple *head-related transfer function* (HRTF) model, $H(\omega, \boldsymbol{x})$ [15]; $sound(\boldsymbol{x}, t) = \sum_{k=1}^{r} |H(\omega_k, \boldsymbol{x})| \, |p_k(\boldsymbol{x})| \, q_k(t)$. Graphics frames were rendered using Pixar's RenderMan software. All floating point computations were performed using double precision.

EXAMPLE (Cymbal): We modeled a large ride cymbal (50cm diameter, bronze), which is known to be a challenging example for modal vibrations [22]. The linear modal model of a ride cymbal produces a very clean tone that sounds more like a smaller crash cymbal, and it is unable to produce the proverbial "crash" sound as well as the nonlinear model. Both models sound very plausible with acoustic transfer.

EXAMPLE (Trash Can with Lid): The nonlinear modal model produces a dramatic improvement in the sound of the trash can (and lid) relative to the linear modal model; the nonlinear model produces a characteristic "crashing" sound, whereas the linear model makes a "ding" sound. The trash can also has very interesting acoustic transfer functions; its FFAT maps reveal intricate structure, partly due to the trash can's side-reinforcing ribs, and very loud values near its opening (see Figure 3.13). The spolling (spinning & rolling) of the trash can lid has a more distinctive sound than the linear model.

EXAMPLE (Water Bottle): We modeled a round 5-gallon water bottle out of polycarbonate plastic, and tuned damping parameters by comparing to informal experiments.

Figure 3.9: **Multibody collision scenarios** were simulated for (from left to right) a cymbal with metal balls, multiple cymbals, two trash cans, a trash can and lid, and polycarbonate water bottles—as well as Figure 3.1.

The nonlinear sound model captures a characteristic drum-like fluttering after impact better than the linear sound model. The complex structure of the FFAT maps are shown in Figure 3.12. A comparison to a real water bottle impact experiment is provided in the accompanying video, and produces a qualitatively similar sound.

EXAMPLE (Plastic Recycling Bin): While less dramatic than other examples, the nonlinear model captures a familiar "wobbling" sound which is missing from the linear model.

MULTIBODY EXAMPLES: We simulated several multibody collision scenarios to demonstrate the feasibility of "Harmonic Shells" for computer animation (see Figure 3.9, and video results).

Stability: Unlike linear modal models which can be stably integrated with IIR filters, our nonlinear subspace vibration model can suffer time-stepping instabilities. Fortunately, subspace integrators are typically more stable than their unreduced counterparts [76]. We observed that our explicit subspace Newmark integrator was stable at audio rates (44.1 kHz) for all examples, except the cymbal which we integrated at 88.2 kHz. In contrast, traditional explicit Newmark integration required an exceedingly small timestep to be stable, e.g., the water bottle required 11.025 MHz rates (or 250$\times$ the 44.1 kHz rate).

COMPARISON (reduced vs. unreduced simulation): For validation, we compared

water bottle impact sounds from our reduced-order model ($\Delta t = 1/44100s$) to those of nonlinear vibrations simulated in a full, unreduced setting via an explicit Newmark integrator ($\Delta t = 1/11025000s$ for stability)—implicit Newmark (with full Newton solves) was less competitive in our experiments. Although the unreduced model produced richer tones at higher amplitude impacts, both sounds were comparable and more interesting than pure linear vibrations. Unfortunately, while the reduced-order model took roughly 17.1 minutes to compute 1 second of sound (1026$\times$ slower than real time), the unreduced approach took 89.8 hours per second of sound (323,000$\times$ slower than real time).

Details of unreduced computation: Given the unreduced displacement $\mathbf{u}$ of the object, modal amplitudes are obtained via projection with the basis $\mathbf{q} = \mathbf{U}^T\mathbf{u}$ so that any of the previously discussed radiation methods may be applied. Given that the simulated model is unconstrained, we take steps to avoid rigid body motions in the unreduced simulation as these will result in errors in the modal projection. A $N \times 6$ "rigid basis" matrix $\mathbf{U}_R$ is constructed out of the rigid modes (those corresponding to eigenvalue 0) computed in equation 2.14. This is used to produce a 6×6 "rigid mass" matrix $\mathbf{M}_R = \mathbf{U}_R^T\mathbf{M}\mathbf{U}_R$. Given the external forces acting at each time step, the component of acceleration resulting in rigid motion is identified as $\mathbf{U}_R\mathbf{M}_R^{-1}\mathbf{U}_R^T\mathbf{f}_{ext}$ and subtracted from the total acceleration vector. This allows the mesh to vibrate freely in place without undergoing rigid translation and rotation.

COMPARISON (different cubature errors): We simulated nonlinear modal models with different cubature errors to informally demonstrate their respective sound behavior. See the video for a comparison of the trash can simulated with cubature errors (and timestep costs) of 15.3% (10.5ms), 10.3% (16.1ms), 6.1% (27ms), and using brute-force subspace integration [76] we simulated 0% (166.7ms). We find that even cubature schemes with large relative error provide a significant qualitative improvement in sound

Model	Mode	Freq (Hz)	kL	θ Resolution (T)	Average Relative Error (%)				Median Relative Error (%)			
					$M=1$	$M=2$	$M=3$	$M=4$	$M=1$	$M=2$	$M=3$	$M=4$
Trash Can	0	71	0.90	18	1.47	6.73	0.95	1.04	0.54	7.2	0.91	1.03
	50	1880	25.82	93	19.51	7.11	1.28	0.69	9.14	3.40	0.67	0.44
	100	2823	38.79	132	44.44	28.36	20.00	5.62	31.26	12.88	9.27	2.66
	150	3698	50.81	168	67.21	35.20	8.27	1.76	29.18	12.22	4.37	0.63
	199	4433	60.80	198	53.37	42.64	17.05	3.14	28.64	16.94	9.56	1.27
Trash Lid	0	112	1.13	20	3.12	8.47	1.36	0.78	2.52	9.04	.14	0.78
	50	2296	23.13	110	36.12	21.91	6.41	1.74	34.89	21.18	5.64	1.64
	100	3997	40.27	180	51.54	24.36	18.07	2.36	51.76	21.22	16.10	1.62
	150	5736	57.79	252	14.09	3.96	1.32	0.47	8.53	2.96	0.75	0.32
	199	6791	68.22	295	60.40	23.39	31.04	2.94	62.28	12.82	27.17	1.39
Water Bottle	0	116	0.98	20	251.35	16.00	1.90	0.032	264.86	11.37	1.67	0.018
	75	1321	11.13	70	26.86	11.80	2.98	0.54	13.07	3.51	1.49	0.35
	150	2197	18.51	106	11.14	6.69	0.86	0.16	8.30	5.77	0.30	0.10
	225	2906	24.49	135	14.03	8.41	1.33	0.31	8.73	4.99	0.65	0.25
	299	3593	30.27	164	32.26	15.84	3.33	0.45	15.60	8.15	1.34	0.26
Recycling Bin	0	62	0.70	18	10.85	3.53	0.40	0.43	6.47	3.46	0.32	0.35
	75	820	9.17	49	13.66	3.79	1.05	0.65	7.45	1.73	0.64	0.34
	150	1329	14.85	70	8.54	3.72	0.87	0.65	6.52	2.30	0.67	0.49
	225	1791	20.01	89	7.36	2.89	0.70	0.35	5.03	1.37	0.46	0.24
	299	2209	24.68	107	12.67	5.03	2.07	0.54	9.13	3.05	12.10	0.36

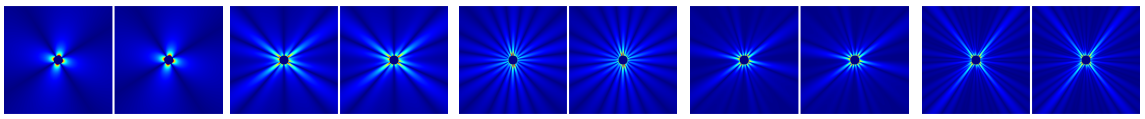
Table 3.3: **Comparison of FFAT map to fast multipole solver** $|p(x)|$ pressure values illustrate that very low relative errors ($\approx 1\%$) can be achieved using a 4-term FFAT map expansion. Errors were computed for representative raster images (see Figure 3.11 for a specific example).

quality over the linear model. Our cubature schemes are chosen to provide a tradeoff between sound quality and evaluation speed.

COMPARISON (with/without energy limitation): To demonstrate pitch glide, we artificially increase the magnitude of forces acting on the water bottle by a factor of 5. See the video for a comparison of this scenario with and without the impulse limiter (§3.3).

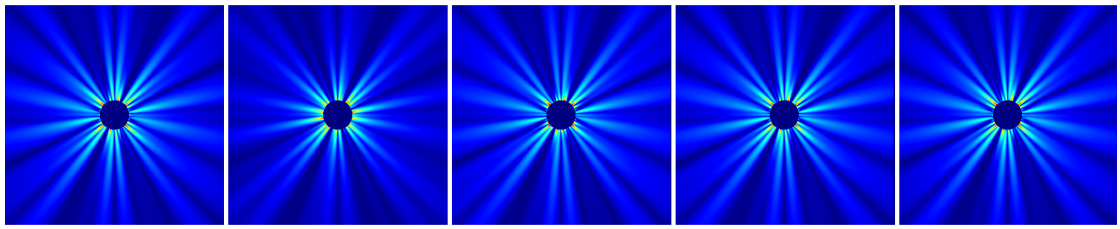
COMPARISON (FFAT vs Fast-Multipole-Method Error): Please see Table 3.3 and Figure 3.10 for FFAT map accuracy demonstrations. Our video provides animated comparisons: FastBEM required $\sim 17h15m$ to synthesize all-mode transfer for the trash-can animation, and $\sim 13h13m$ for the water bottle animation; for both animations, FFAT Map evaluation required under a second.

COMPARISON: Different FFAT map expansion orders, M**,** are shown in Figure 3.11 for the highest frequency mode of the trash can. We use at most 4-maps/mode



Mode 0 (71 Hz) Mode 50 (1880 Hz) Mode 100 (2823 Hz) Mode 150 (3698 Hz) Mode 199 (4433 Hz)

Figure 3.10: **Comparison of sound pressures,** $|p(\boldsymbol{x})|$, between BEM (left) and FFAT map approximations (right) for various "trash can" modes. In all cases, the 4-term FFAT map ($< 6\%$ average error) results both look and sound essentially same. Error values are given in Table 3.3.



Exact $M=1$ (53.4%) $M=2$ (42.6%) $M=3$ (17.1%) $M=4$ (3.1%)

Figure 3.11: **Comparison of FFAT maps of different order** for the trash can (mode 199). (Far Left) Exact transfer field evaluated using the fast multipole method. The remaining figures ($M = 1 \ldots 4$) show the result of optimizing the FFAT models with different M values (# maps $= M$), and their average pointwise relative errors for the $|p(\boldsymbol{x})|$ rasters. The single-term approximation would provide enough accuracy for real-time applications using linear modal models.

($M=4$) in all of our rendered examples. Convergence is obtained for increasing M values; an error analysis is provided in Table 3.3. Please see the video for comparisons; in practice, similar sounds are obtained for all M values, suggesting that even one texture map ($M=1$) is sufficient.

3.6 Conclusion

We have presented a practical method for generating plausible impact sounds for thin shells. By leveraging reduced-order modeling, we can produce nonlinear modal models that enable simulation of hundreds of vibration modes with fully coupled nonlinear modal dynamics. We proposed a method to limit impact magnitudes and overcome

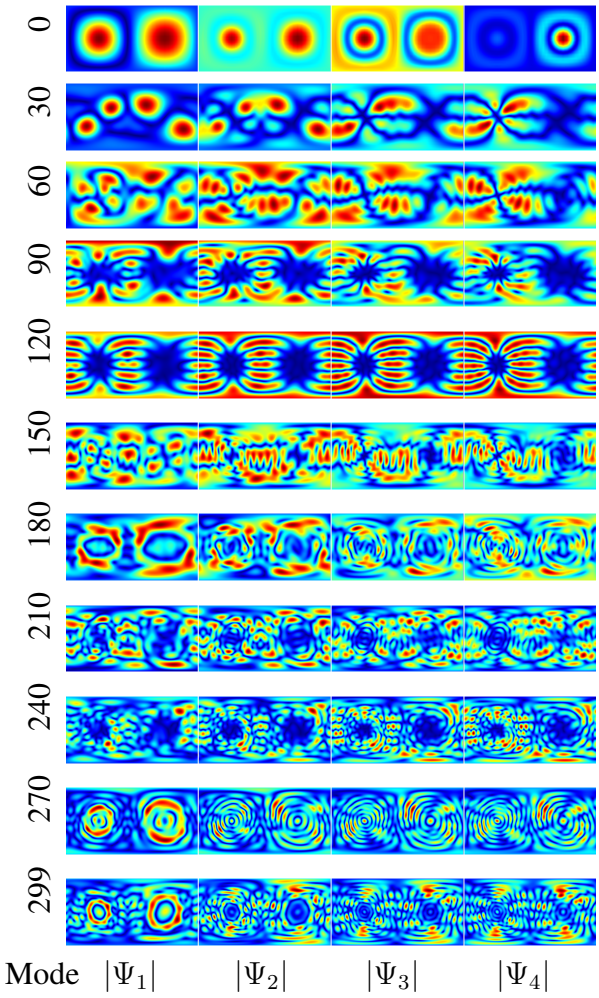


Figure 3.12: **FFAT Maps (Water Bottle)**

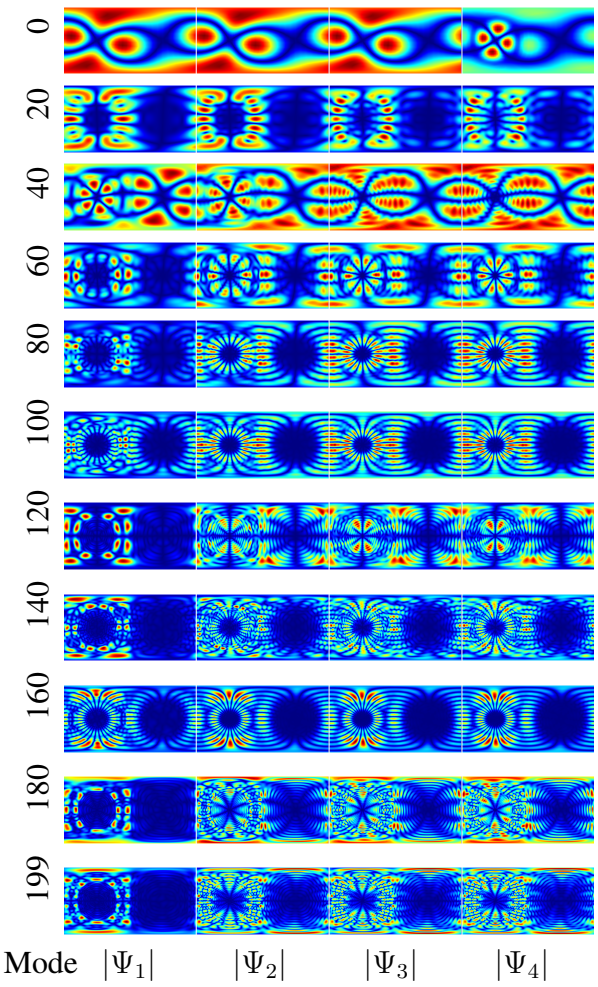


Figure 3.13: **FFAT Maps (Trash Can)**

pitch-glide artifacts. Compared to linear modal sound models, our objects produce more characteristic "crashing" and "rumbling" sounds.

We also proposed *FFAT maps*, a fast texture-based approximation of each mode's far-field acoustic transfer function that captures complex spatial structure, and are more generally applicable than to just thin shells. They exploit the observation that transfer functions exhibit complex angular structure, but possess strong radial coherence. By using an accurate transfer solution, e.g., from a fast multipole solver, we produced far-field acoustic transfer map approximations that enable fast run-time evaluation of the transfer function to very low tolerances (e.g., 1%). In practice, we found that even a

single FFAT map texture per mode ($M = 1$) produced excellent results.

Limitations and Future Work: There are numerous ways to improve this result in future work. Simulating the full range of audible "all-frequency" sound poses numerous challenges, not only for precomputing thousands of vibration modes but also for coupling them together. The $O(r^2)$ complexity of the nonlinear modal model reflects the intrinsic complexity of simulating r coupled modes, but a near-linear-time subspace force algorithm would be a big breakthrough for all-frequency nonlinear sound synthesis, especially since the modal amplitudes are needed for transfer-based rendering. In all cases, we would have liked to have used more vibration modes to produce "all-frequency" sound renderings; large models can also require many modes. The shell-based cubature schemes exhibit slower convergence rates than volumetric models [3], and higher accuracy and more scalable methods are required for faster and/or more complex models. We use a simplified rigid-body contact model, but collision processing should account for object vibrations (ideally in a reduced-order manner [72]) to properly capture chattering.

Beyond thin shells, how to devise efficient methods for evaluating all-frequency nonlinear vibration-based sound is an open problem. Modal locking reflects the limitations of the linear modal shape model, and we need more expressive shape models to handle difficult nonlinear and noise-like phenomena; a fully developed (chaotic) cymbal crash is currently beyond the capability of our reduced-order vibration model, although the cymbal's acoustic transfer model appears plausible. Preliminary experiments with thin-shell models using nonlinear shape functions based on modal warping did not provide a significant improvement [26]. More generally, we need methods to evaluate accurate sound for large-deformation animations. Buckling is also a challenging nonlinear phenomenon which can produce significant sound radiation, e.g., crumpling paper.

Our FFAT maps can provide high-accuracy acoustic transfer, but would benefit

from precomputation techniques for improved spatial sampling, estimation, and texture compression—FFAT maps can be "big." By construction, our current FFAT models are less accurate for near-field listening positions, which may be a problem for some applications. Far-field listening positions should also include time delay effects, which can be complicated for dynamic objects. Nonlinear vibrations can introduce other significant frequency contributions in to each mode's vibration especially for high-frequency modes (see Figure 3.8), so more complex nonlinear radiation models are needed at high frequencies—multi-frequency mode radiation models may provide more realistic sound. We have only considered single-object sound transport to leverage precomputation, and it still remains to include multi-object and environment scattering for correct auralization.

Perceptually based rendering methods are desperately needed to strike a favorable balance between model accuracy and simplicity. For example, our thin-shell transfer maps exhibit increasingly complex structure at high frequencies and are extremely difficult to compute without sophisticated fast multipole solvers. Arguably it is hard to hear all of this structure in sound renderings, and we therefore desire perceptually based precomputation and rendering techniques which display only what is necessary, and avoid computing what is not.

CHAPTER 4

PRECOMPUTED ACCELERATION NOISE FOR IMPROVED RIGID-BODY SOUND

Like Chapter 3, this chapter will also describe a method for addressing limitations of the linear modal sound algorithm from §2.2.1. However, rather than presenting a method for synthesizing sound from object vibrations, we turn our attention to *acceleration noise* – sound produced when an object experiences abrupt rigid-body acceleration due to, for instance, collisions or other contact events. We approach the problem of acceleration noise synthesis in two main steps. First, we estimate continuous contact force profiles from rigid-body impulses using a simple model based on Hertz contact theory. Next, we compute solutions to the acoustic wave equation due to short acceleration pulses in each rigid-body degree of freedom. We introduce an efficient representation for these solutions – *Precomputed Acceleration Noise* – which allows us to accurately estimate sound due to arbitrary rigid-body accelerations. We find that the addition of acceleration noise significantly complements the standard modal sound algorithm, especially for small objects.

4.1 Introduction

The rattling of coins dropped on a table, the jingling of a set of car keys, and the cascade of noise from a shattering pane of glass are all familiar sound phenomena. Rigid-body dynamics simulation for scenarios such as these is a widely studied field in the computer animation community. Physically based rigid-body solvers are capable of producing detailed visual behavior virtually indistinguishable from reality. The linear modal sound model (§2.2.1) is the most popular approach for resolving the sound of colliding rigid bodies in scenarios like these [129, 100, 12]. This method efficiently and accurately

Figure 4.1: **Precomputed Acceleration Noise:** Our model efficiently synthesizes acceleration noise due to rigid-body collisions in a variety of multibody collision scenarios.

captures the collision-induced ringing noise produced by a wide variety of objects. Numerous methods for evaluating *acoustic transfer* functions – solutions to the Helmholtz equation characterizing how vibrating objects produce sound – have been combined with modal sound algorithms to produce more realistic results (e.g., [70] and the FFAT map approach from Chapter 3).

In spite of these advances, there are many rigid-body collision scenarios for which current sound models produce unconvincing results. Colliding rigid-body objects produce sound primarily due to two sources: "ringing noise" [110] and "acceleration noise" [109]. Ringing noise refers to sound from object vibrations. This is the type of sound which is handled by the methods from Chapter 3. Acceleration noise is produced when objects undergo rapid rigid-body accelerations. If a body experiences acceleration over a sufficiently short time scale, the resulting pressure disturbance in the surrounding medium is perceived as sound. While current rigid-body sound models synthesize convincing ringing noise (§2.2 and Chapter 3), no efficient models exist for synthesizing sound due to acceleration noise. Consequently, synthetic rigid-body impact sounds tend to have an incorrect initial attack and lack the "crispness" characteristic of real impact sounds. In fact, for small objects that only vibrate at frequencies beyond the range of human hearing (small ball bearings, dice, etc.), the approach discussed in §2.2 produces no sound whatsoever. While this omission is obvious even for certain single-object collision sounds, it is particularly noticeable in scenes involving large ensembles of small colliding objects. Prior work on modal sound synthesis has modeled transient contact

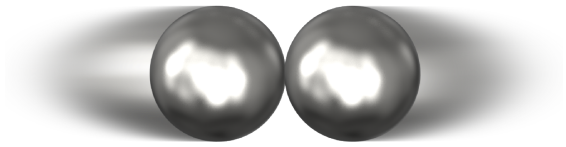


Figure 4.2: Small ball bearings vibrate too quickly to produce audible modal sound. Instead, their sound is the result of rigid acceleration experienced during collisions.

sounds based on recorded audio [129, 84, 138]. Different approaches for improving contact modeling in modal sound synthesis have been studied in [107] and [138], but these methods still omit transient acceleration noise. Verma and Meng [130] introduced transient-modeling synthesis (TMS), which extends earlier work on spectral modeling synthesis [116]. TMS provides a general system for decomposition and synthesis of audio signals involving both sinusoidal (modal) and transient components; however, it does not provide a physics-based approach for synthesizing sounds from arbitrary rigid objects.

To address this limitation, we propose a simple and efficient model for acceleration noise which can be easily integrated with existing rigid-body sound pipelines. Resolving acceleration noise for a rigid-body simulation is accomplished via two main steps:

1. *Contact force estimation*: Rigid-body solvers separate colliding objects by applying repulsive impulses to produce instantaneous velocity changes in the colliding objects. We use Hertz contact theory [62, 73] to estimate physically plausible continuous rigid-body acceleration profiles from each collision impulse.
2. *Acoustic radiation*: We present an efficient precomputed representation for sound due to objects undergoing short acceleration pulses. This allows us to reconstruct arbitrary acceleration noise signals from an object, while avoiding impractical audio-rate time stepping of the acoustic wave equation.

Other Related Work: Acceleration noise has been studied previously outside of the graphics community, e.g., for noise control requirements in industrial settings. Koss and Alfredson [75] investigated the transient sound radiated by colliding spheres and found that a contact model based on Hertz contact theory could be used to approximately recover pressure waveforms measured in experiments. Other studies have investigated acceleration noise due to colliding spheres [109, 110] and other simple geometries [40, 135]. Acceleration noise for impacted plates has been studied experimentally by Wåhlin et al. [132] and Chaigne & Lambourg [23]. Numerical models for plate acceleration noise have been investigated by Shedin et al. [113] using experimentally recorded impact forces, and by Lambourg et al. [77] and Ross & Ostiguy [112] using Hertz-like impact forces. These numerical approaches have shown good agreement with experimental results; however, they use expensive finite element or finite difference discretizations to simulate dynamics, making them undesirable for animation sound purposes.

In principle, both ringing noise and acceleration noise can be resolved simultaneously by explicitly solving for a body's surface motion using standard numerical approaches such as the finite element method. The effect of an object's surface motion can be propagated to the listener by solving the time-domain wave equation in the region surrounding the object (e.g., [90]). In O'Brien et al. [99], the authors used explicit time-stepping of nonlinear elastic finite element meshes to resolve object surface deformations at audio time-stepping rates and computed sound by directly propagating surface velocities at each object triangle to the listener's location. While this method could in principle resolve detailed surface deformations sufficient for synthesizing both ringing and acceleration noise, rigid-body accelerations (and indeed high-frequency ringing noises) are the result of compression occurring over very small regions and time scales [73]. Accurately resolving these scales via finite element simulation requires spatial and temporal resolutions to be prohibitively high for computer animation purposes. More-

over, we find that rigid-body acceleration tends to produce complex time-varying pressure signals even for simple pulse-like velocity changes (see Figures 4.4, 4.5). Producing sound directly from an object's surface velocity fails to capture this wave radiation.

Hertz originally proposed a model for normal contact between elastic bodies based on a nonlinear spring force [62]. This model can be used to determine the continuous contact forces acting on dynamically colliding bodies [73, 46]. Analytical approximations of these time-dependent contact forces are also available for certain geometries. Hertz contact and other similar theories are used widely in the multibody systems community to predict continuous contact forces [53, 114, 45, 46, 47]. In the context of sound synthesis, Hertz-like contact force models have been considered for excitation of modal vibration in the computer graphics community [129] and elsewhere [122].

4.2 Background

As in §2.1.1, we consider an object O along with its exterior domain Ω and the domain boundary $\partial\Omega$ (the surface of O) with normal field $\mathbf{n}(\boldsymbol{x})$ defined to be pointing in to Ω. Acoustic pressure fluctuations in Ω are modeled by the acoustic wave equation (2.1) with boundary condition (2.2).

We may also express (2.1) as a pair of first-order partial differential equations (PDEs) describing acoustic pressure $p(\boldsymbol{x})$ and particle velocity $\mathbf{v}(\boldsymbol{x})$:

$$\frac{\partial p(\boldsymbol{x},t)}{\partial t} = -c^2\rho\nabla\cdot\mathbf{v}(\boldsymbol{x},t) \tag{4.1}$$

$$\frac{\partial \mathbf{v}(\boldsymbol{x},t)}{\partial t} = -\frac{1}{\rho}\nabla p(\boldsymbol{x},t) \tag{4.2}$$

In this case, (2.2) and (4.2) imply that $\frac{\partial v_n}{\partial t} = a_n$ on $\delta\Omega$, where v_n is the normal particle velocity on the object's surface. The wave equation boundary condition (2.2) accounts

for contributions from arbitrary motion by object O's surface, including rigid-body motion. Intuitively, when an object moves through the air, the velocity of the surrounding air must change to match that of the moving object at its surface. When an object undergoes an abrupt change in velocity, the resulting fluctuation in the surrounding air pressure is propagated according to the wave equation and interpreted by a listener as sound (see Figure 4.5).

Most prior approaches for synthesizing sound from vibrating objects have assumed that the visual motion of an object is governed entirely by rigid-body dynamics. Object deformations are assumed to be small and independent of an object's rigid-body behavior. Deformations are modeled by simulating vibrations in small basis of linear mode shapes, transformed in to the object's rigid-body frame (§2.2, Chapter 3, [100], etc.). Under the assumption that rigid motion and modal vibration are independent, we may write the normal acceleration on an object's surface as $a_n(\boldsymbol{x}, t) = a_n^R(\boldsymbol{x}, t) + a_n^M(\boldsymbol{x}, t)$ where a_n^R and a_n^M refer to normal surface acceleration due to rigid-body motion and modal vibration, respectively. By the linearity of (2.1) and (2.2), we can write the acoustic pressure as $p(\boldsymbol{x}, t) = p^R(\boldsymbol{x}, t) + p^M(\boldsymbol{x}, t)$ where p^R and p^M are the pressure contributions due to rigid-body motion and modal surface vibration. Existing rigid-body sound models only compute p^M, neglecting the acceleration noise contribution p^R. This omission is particularly noticeable for small, hard objects such as plastic dice and steel ball bearings. The linear vibration modes for these objects (see §2.2.1) vibrate at frequencies well above 20 kHz. Consequently, modal sound algorithms predict sound well beyond the frequency range of human hearing for these objects.

4.3 Contact Force Estimation

We run rigid-body simulations using a solver based on the work of Guendelman et al. [58] in which collisions are resolved via *impulses*. Impulses are time-integrated forces that are treated as instantaneous changes in linear and angular momentum for the purpose of physics-based animation of rigid-body dynamics. When two objects collide, impulses are applied to ensure that the resulting velocity changes cause intersecting regions in the objects to separate. This approach of instantaneously updating velocities to resolve contact is widely used in rigid-body animation and produces compelling visual results. Unfortunately, this approach does not provide a suitable continuous acceleration profile for use in (2.2). We could, in principle, acquire continuous acceleration data by resolving rigid-body contacts with a simple Kelvin-Voigt penalty law (used previously for sound synthesis in, e.g., Chapter 3). However, recovering physically plausible acceleration time scales would require both careful parameter tuning and simulation of rigid-body dynamics at very high rates.

Contact force modeling is a widely studied field [73, 46] and numerous continuous contact force models are available. Methods have been developed to account for damping during contact events [68], hysteresis [78] and arbitrary geometric structure in contact regions [63]. Evaluating these contact force models in the context of dynamics simulation requires careful parameter tuning and potentially costly numerical simulation [101]. The goal in this section is to present a simple and efficient model for estimating the continuous contact forces necessary to demonstrate rigid acceleration noise. Therefore, we appeal to a simple model based on the Hertz theory of elastic contact.

4.3.1 Hertz Contact Theory

Hertz contact theory states that the normal contact force between two colliding elastic bodies is given by

$$f = Kd^{1.5} \tag{4.3}$$

where d is the penetration depth at the contact point and K is a constant depending on the material properties and local contact geometry of the colliding bodies. Johnson [73] provides analysis of the contact forces experienced for two colliding spheres. We briefly summarize the key points of this analysis here. For a collision between two frictionless spheres, $K = (4/3)\sqrt{r}E^*$ and the time dependence of the contact force for a collision beginning at time t_0 can be approximated by a half-sine pulse

$$S(t; t_0, \tau) = \begin{cases} \sin\left(\frac{\pi(t-t_0)}{\tau}\right) & \text{if } t_0 \leq t \leq t_0 + \tau \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

where τ is a time scale defined by

$$\tau = 2.87 \left(\frac{m^2}{rE^{*2}V}\right)^{1/5}. \tag{4.5}$$

V is the normal impact speed and the other constants used in (4.5) are defined as follows:

$$\frac{1}{r} = \frac{1}{r_1} + \frac{1}{r_2}, \quad \frac{1}{m} = \frac{1}{m_1} + \frac{1}{m_2}, \quad \frac{1}{E^*} = \frac{1-\nu_1^2}{E_1} + \frac{1-\nu_2^2}{E_2}. \tag{4.6}$$

R_i, m_i, ν_i and E_i are the radius, mass, Poisson ratio and Young's modulus for spheres $i = 1, 2$. Force profiles of the form (4.5) have been used to model continuous contact forces for acceleration noise due to simple geometries in [75, 109, 40, 135].

4.3.2 Contact Time Scale Estimation

Given a collision between objects O_1 and O_2, let $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ be the contact locations in the coordinate frames of O_1 and O_2 (see Figure 4.3(b)). We use (4.4-4.5) to estimate

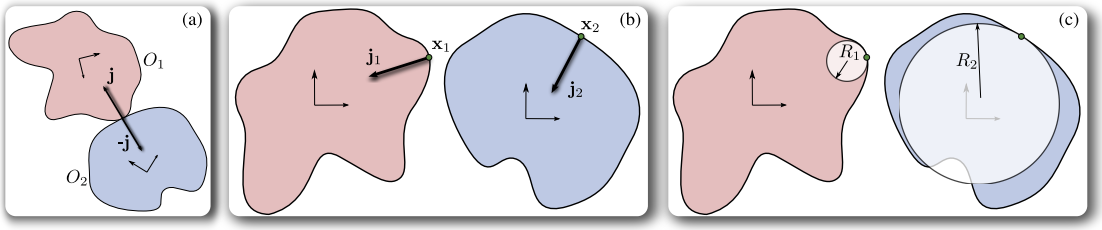


Figure 4.3: **Contact Geometry:** *(a)*: Two objects collide and experience equal and opposite impulses; *(b)*: The impulses and contact positions and sound listening positions are transformed in to each object's rest frame. *(c)*: We fit spheres whose curvatures match the curvatures at points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ on O_1 and O_2. These proxy geometries are used to determine a Hertz impact time scale using (4.5).

a time scale and time-dependent contact force for the collision. Let $\mathbf{n}_i$ be the contact impulse direction in O_i's coordinate frame. To leverage the analytical force profile (4.4-4.5) we locally approximate O_1 and O_2 with spheres in the vicinity of the contact points. We compute the discrete mean curvature $H_1(\boldsymbol{x}_1)$ of O_1 at $\boldsymbol{x}_1$ (similarly, $H_2(\boldsymbol{x}_2)$) using the methods of [91]. Next, we estimate proxy sphere radii as $r_1 = 1/H_1(\boldsymbol{x}_1)$ and $r_2 = 1/H_2(\boldsymbol{x}_2)$. If O_1 and O_2 are in fact spheres, then r_1 and r_2 recover their radii exactly up to mesh discretization error. We use r_1 and r_2 to compute the term $1/r$ from (4.6). The material parameter in (4.6) can be computed trivially using the material properties of O_1 and O_2. Finally, we require the effective mass term $1/m$. For object O_i, let

$$\frac{1}{m_i} = \frac{1}{M_i} + \mathbf{n}_i^T \mathbf{I}_{\boldsymbol{x}_i - \boldsymbol{x}_0^i}^T \mathbf{M}_i^{-1} \mathbf{I}_{\boldsymbol{x}_i - \boldsymbol{x}_0^i} \mathbf{n}_i. \tag{4.7}$$

That is, $1/m_i$ is the inverse effective mass experienced by a force acting on body O_i at position $\boldsymbol{x}_i$ in direction $\mathbf{n}_i$. M_i, $\boldsymbol{x}_0^i$ and $\mathbf{M}_i$ refer to the mass, center of mass and moment of inertia matrix in O_i's coordinate frame. For a vector $\mathbf{z} \in \mathbb{R}^3$, $\mathbf{I}_\mathbf{z}$ is the 3×3 cross product matrix defined by $\mathbf{I}_\mathbf{z}\mathbf{v} = \mathbf{z} \times \mathbf{v}, \forall \mathbf{v} \in \mathbb{R}^3$. The geometry used for this calculation is illustrated in Figure 4.3.

4.3.3 Acceleration Profile Estimation

Following the notation of §4.3.2, suppose that the impulses applied to some position on O_1 and O_2 are $\mathbf{j}$ and $-\mathbf{j}$ in world coordinates (see Figure 4.3(a)). Let $\boldsymbol{x}_1, \boldsymbol{x}_2$ and $\mathbf{j}_1, \mathbf{j}_2$ be the collision points and values of the impulse when transformed in to the respective coordinate frames of O_1 and O_2 and let $\widehat{\mathbf{j}}_i = \mathbf{j}_i / \|\mathbf{j}_i\|$. For an impulse $\mathbf{j}$ occurring at time t_0 and producing a force profile with time scale τ (§4.3.2), we choose a scaling factor γ so that the time-integrated Hertz contact force matches the magnitude of the impulse:

$$\int_0^\infty \gamma S(t; t_0, \tau) dt = \|\mathbf{j}\| . \tag{4.8}$$

From (4.4) and (4.8) we see that $\gamma = \pi\|\mathbf{j}\|/2\tau$. Scaling by γ guarantees consistency with simulated rigid-body dynamics. Finally, we assume that the change in position/orientation of body O_i is negligible over the time period during which the contact force is applied. This assumption is reasonable given the short timescales associated with contact (τ is typically in the range of 10-100μs). Under this assumption, the translational and angular accelerations in O_i's coordinate frame are given by

$$\mathbf{a}_i(t) = \frac{\gamma}{M_i} \widehat{\mathbf{j}}_i \, S(t; t_0, \tau), \tag{4.9}$$

$$\boldsymbol{\alpha}_i(t) = \gamma \mathbf{M}^{-1} \left(\left(\boldsymbol{x}_i - \boldsymbol{x}_0^i \right) \times \widehat{\mathbf{j}}_i \right) S(t; t_0, \tau). \tag{4.10}$$

For an arbitrary rigid-body collision, we use (4.9-4.10) to determine continuous translational and rotational acceleration signals with time-dependence given by a half-sine pulse (4.4).

4.4 Precomputed Acceleration Noise

In this section, we introduce an efficient method for synthesizing sound due to continuous rigid-body acceleration.

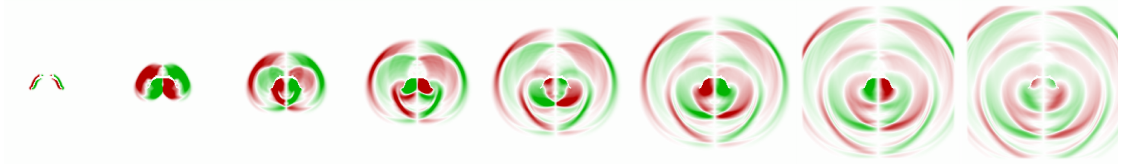


Figure 4.4: **Rigid acceleration pressure:** Time evolution of the pressure field surrounding a bowl undergoing a short horizontal acceleration pulse. We precompute these rigid-body acceleration responses for sound synthesis.

4.4.1 Directional Pressure Fields

Let $\mathbf{a}(t) = [a_1(t)\ a_2(t)\ a_3(t)]^T$, $\boldsymbol{\alpha}(t) = [\alpha_1(t)\ \alpha_2(t)\ \alpha_3(t)]^T$ and $\boldsymbol{x}_0$ refer to the translational acceleration, angular acceleration and center of mass position of a rigid body O at time t in O's coordinate frame. Normal rigid surface acceleration a_n^R at position $\boldsymbol{x}$ on O's surface can be written as

$$\begin{aligned} a_n^R(\boldsymbol{x},t) &= \mathbf{a}(t)\cdot\mathbf{n}(\boldsymbol{x}) + (\boldsymbol{\alpha}(t)\times(\boldsymbol{x}-\boldsymbol{x}_0))\cdot\mathbf{n}(\boldsymbol{x}) \\ &= \sum_{i=1}^{3} a_i(t)\mathbf{e}_i\cdot\mathbf{n}(\boldsymbol{x}) + \sum_{i=1}^{3}\alpha_i(t)\,(\mathbf{e}_i\times(\boldsymbol{x}-\boldsymbol{x}_0))\cdot\mathbf{n}(\boldsymbol{x}) \end{aligned} \tag{4.11}$$

where $\mathbf{e}_i \in \mathbb{R}^3$ is the vector with components $e_{ij} = \delta_{ij}$. We write rigid-body acceleration noise in O's coordinate frame as a sum of contributions from each of these acceleration components:

$$p(\boldsymbol{x},t) = \sum_{i=1}^{3} p_{T,i}(\boldsymbol{x},t) + \sum_{i=1}^{3} p_{R,i}(\boldsymbol{x},t). \tag{4.12}$$

The components $p_{T,i}$ are contributions due to translational acceleration; solutions to (6.2) subject to

$$\nabla p_{T,i}\cdot\mathbf{n}(\boldsymbol{x}) = -\rho\, a_i(t)\,\mathbf{e}_i\cdot\mathbf{n}(\boldsymbol{x}) \tag{4.13}$$

and $p_{R,i}$ are contributions due to rotational acceleration; that is, solutions to (6.2) subject to

$$\nabla p_{R,i}\cdot\mathbf{n}(\boldsymbol{x}) = -\rho\,\alpha_i(t)\,(\mathbf{e}_i\times(\boldsymbol{x}-\boldsymbol{x}_0))\cdot\mathbf{n}(\boldsymbol{x}). \tag{4.14}$$

Figure 4.4 illustrates the time evolution of $p_{T,1}$ for a bowl undergoing a short acceleration pulse along its x-axis. Given a continuous rigid acceleration signal, we can solve (4.1-

4.2) for each of the boundary conditions (4.13) and (4.14) independently and use (4.12) to recover the total acceleration noise. It is straightforward to time step (4.1-4.2) using a staggered grid finite difference discretization (e.g., [82]). However, the cost of this approach makes it infeasible for animation sound synthesis.

We can simplify (4.12-4.14) somewhat by introducing the acceleration vector

$$\mathbf{z}(t) = [a_1(t)\ a_2(t)\ a_3(t)\ \alpha_1(t)\ \alpha_2(t)\ \alpha_3(t)]\,. \tag{4.15}$$

Using (4.15) we can rewrite (4.12) as

$$p(\boldsymbol{x}, t) = \sum_{i=1}^{6} p_i(\boldsymbol{x}, t) \tag{4.16}$$

where

$$\nabla p_i(\boldsymbol{x}, t) \cdot \mathbf{n}(\boldsymbol{x}) = z_i(t)\mathbf{g}_i(\boldsymbol{x}) \tag{4.17}$$

and

$$\mathbf{g}_i(\boldsymbol{x}) = \begin{cases} -\rho \mathbf{e}_i \cdot \mathbf{n}(\boldsymbol{x}) & \text{if } 1 \leq i \leq 3 \\ -\rho(\mathbf{e}_{i-3} \times (\boldsymbol{x} - \boldsymbol{x}_0)) \cdot \mathbf{n}(\boldsymbol{x}) & \text{if } 4 \leq i \leq 6 \end{cases}. \tag{4.18}$$

4.4.2 Precomputed Acceleration Noise

Instead of directly solving (4.1-4.2) over the length of a rigid-body animation, we precompute solutions for short acceleration pulses in each of the 6 rigid-body degrees of freedom. We use these pulses to interpolate the boundary conditions (4.17) and reconstruct acceleration sound due to arbitrary rigid-body acceleration.

For an object O we define a pulse time scale h and pulse function $\psi(t; h)$. We use a Mitchell-Netravali cubic filter [92]

$$\psi(t; h) = \frac{1}{18} \begin{cases} -15y^3 + 18y^2 + 9y + 2 & |t| \leq h \\ 5\,(1+y)^3 - 3\,(1+y)^2 & h \leq |t| \leq 2h, \\ 0 & \text{otherwise,} \end{cases} \tag{4.19}$$

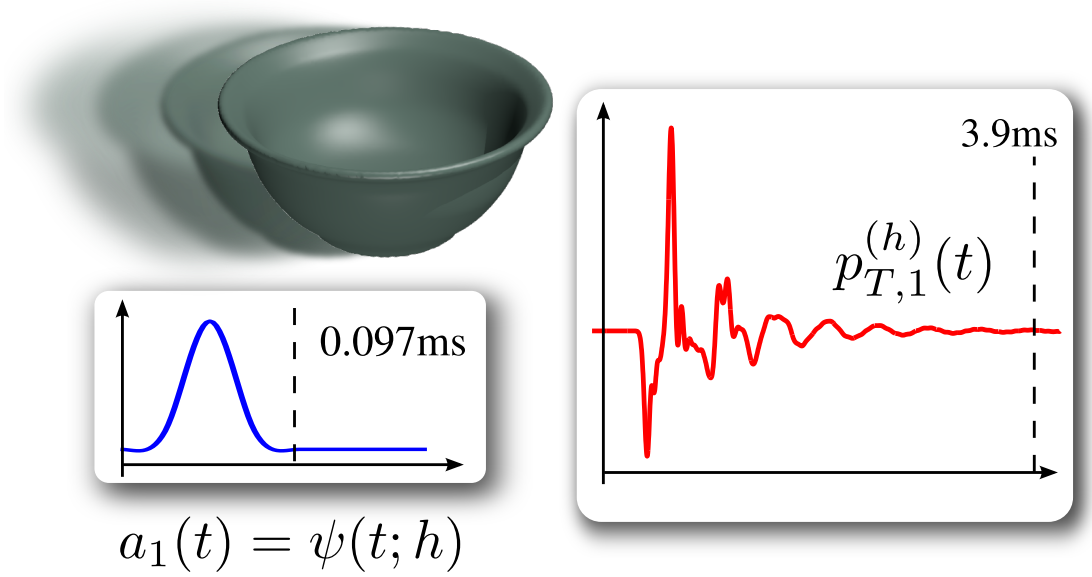


Figure 4.5: **Precomputing directional pulses:** We accelerate the bowl along its x-axis with an acceleration profile given by $\psi(t;h)$. This acceleration produces the pressure fluctuations $p_{T,1}^{(h)}(t)$ at a listening position $\boldsymbol{x}$ exterior to the object. We observe that pressure fluctuations at $\boldsymbol{x}$ persist for much longer than the duration of $\psi(t;h)$ due to acoustic reflections inside the bowl.

where $y \equiv 1-|t|$. Let $p_i^{(h)}(\boldsymbol{x},t)$ be the solution to (4.1-4.2) for $t \geq -2h$ subject to (2.14) with $a_i(t) = \psi(t;h)$. Figure 4.5 illustrates the boundary condition (4.17) for $i = 1$ and the resulting pressure time series $p_1^{(h)}(\boldsymbol{x},t)$ $\left(p_{T,1}^{(h)}\right)$ at an exterior listening position. For arbitrary accelerations $a_i(t)$ and $\alpha_i(t)$ defined for $t \geq 0$, the directional pressure fields are approximated by

$$p_i(\boldsymbol{x},t) \approx \sum_{k=0}^{\infty} z_i(kh)\, p_i^{(h)}(\boldsymbol{x}, t-kh) \tag{4.20}$$

which follows from the linearity of (4.1-4.2) and the approximation of boundary condition (4.17) by basis functions $\psi(t;h)$.

The functions $p_i^{(h)}$ discussed above represent the sounds produced by an object experiencing a short acceleration pulse in each of its six rigid-body degrees of freedom. Assuming that we are able to compute these functions at arbitrary positions/times, (4.20) allows us to reconstruct sound due to arbitrary rigid-body accelerations (up to a temporal

resolution determined by h). Unfortunately, computing $p_i^{(h)}$ for every listening time and position required by an animation is impractical. Therefore, we introduce a data-driven representation for the fields $p_i^{(h)}$ and by discretizing the angular space surrounding an object and computing a radial series approximation of the pressure field in each direction. This representation is similar to the far-field acoustic transfer (FFAT) maps introduced in Chapter 3, in which we compute radial series approximations to solutions of the Helmholtz equation. However, FFAT maps do not handle time-dependent solutions, and cannot be directly used here.

The remainder of this section details our representation for the functions $p_i^{(h)}$. Since we apply identical methods to each of these 6 functions, we will omit subscripts and superscripts and refer to the function to be approximated simply as $p(\boldsymbol{x}, t)$ for the remainder of this section. Let (R, θ, ϕ) be the spherical coordinates of a listening position $\boldsymbol{x}$ relative to O's center of mass. We model the pressure field at $\boldsymbol{x}$ as

$$p(\boldsymbol{x}, t) = \sum_{k=1}^{N} \frac{1}{R^k} \, q_k \left(\theta, \phi, t - \frac{R}{c} \right) \tag{4.21}$$

for some number of terms N, where c is the speed of sound. For a fixed angular direction (θ, ϕ) we discretize the signals q_k at sampling rate $1/\Delta t$ and define

$$\left\{ \ldots, q_k^{-2}, q_k^{-1}, q_k^0, q_k^1, q_k^2, q_k^3, \ldots \right\} \text{ where } q_k^j = q_k(\theta, \phi, j\Delta t). \tag{4.22}$$

The continuous signal q_k is computed via

$$q_k(\theta, \phi, t) = \sum_j q_k^j \, \psi(t - j\Delta t; \Delta t) \tag{4.23}$$

where ψ is the cubic filter (4.19). To determine the numerical values of q_k^j, we precompute the following pressure time series

$$\left\{ \ldots, p_i^{-2}, p_i^{-1}, p_i^0, p_i^1, p_i^2, p_i^3, \ldots \right\} \text{ where } p_i^\ell = p(\theta, \phi, R_i, \ell\Delta t) \tag{4.24}$$

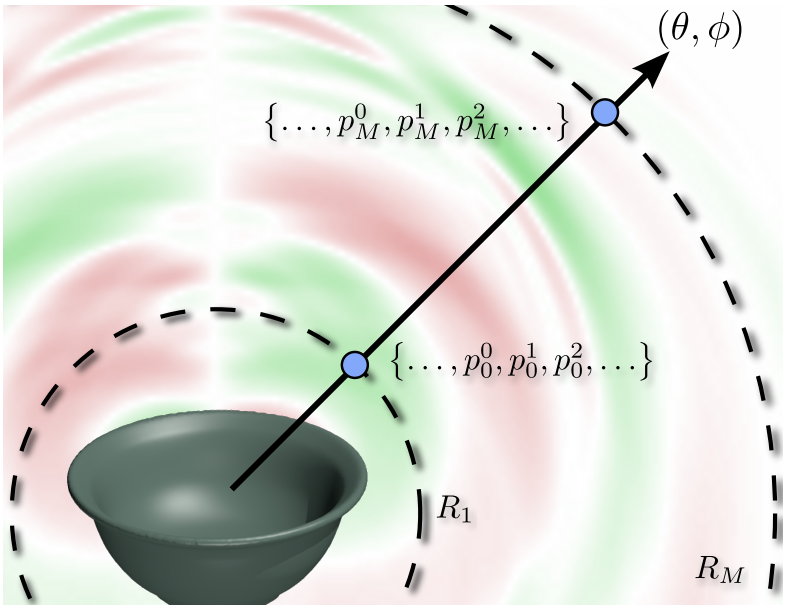


Figure 4.6: **Precomputation Sampling Geometry:** The pressure series (4.24) are computed at radii $R_1, \ldots R_M$ in the direction (θ, ϕ) via direct numerical simulation of the acoustic wave equation. The innermost and outermost radii are visualized and the points at which pressure series are evaluated are shown in blue.

for several radii $R_1, \ldots, R_M$ with $M \geq N$. Then, using (4.21) and (4.23) we have

$$p_i^\ell = \sum_{k=1}^{N} \frac{1}{R_i^k} q_k \left(\theta, \phi, \ell\Delta t - \frac{R_i}{c}\right) \tag{4.25}$$

$$= \sum_{k=1}^{N} \sum_j \frac{\psi\left((\ell - j)\Delta t - \frac{R_i}{c}\right)}{R_i^k} q_k^j. \tag{4.26}$$

We write (4.26) as a system of linear equations $\mathbf{Aq} = \mathbf{p}$ and use a truncated singular value decomposition (TSVD) to find a least-squares solution. Given the values q_k^j provided by this solution, $p(\boldsymbol{x}, t) = p(\theta, \phi, R, t)$ can be evaluated for arbitrary R and t using (4.21) and (4.23). Finally, we uniformly discretize the angle space around $\boldsymbol{x}_0$ and repeat this procedure for each direction, noting that the system (4.26) is the same for all directions, so its SVD only needs to be computed once.

For each of our example objects we precompute $N = 2$ terms using solutions computed at $M = 5$ radii. We define $R_1 = 2R$ where R is the radius of object O's bounding sphere centered at $\boldsymbol{x}_0$. For subsequent radii, we use $R_i = \kappa^{i-1} R_1$. We choose $\kappa = 2^{1/4}$

so that $R_5 = 4R$. Outgoing directions from each object are discretized using uniform 40×80 or 5×10 discretizations of the (θ, ϕ) angle space (see §4.5 for discussion). We evaluate the field in arbitrary directions using linear interpolation in (θ, ϕ) angle space.

4.4.3 Synthesizing Rigid Acceleration Noise

We combine the time-scale estimation described in §4.3 with precomputed acceleration noise to efficiently synthesize rigid-body acceleration sound. The time-scale h (§4.4.2) is determined via a simple approach based on (4.5). Let m_1, ρ_1, E_1 and ν_1 be the mass, density, Young's modulus and Poisson ratio of object O. Next, let $r_1 = (0.75m_1/\rho_1)^{1/3}$; that is, the radius of the sphere whose mass and density match those of O. Using (4.5), we compute the time scale τ due to the collision of this sphere with another sphere of infinite radius, mass and stiffness $r_2 = m_2 = E_2 = \infty$ (effectively an infinitely stiff, infinitely massive plane) at $V = 5$m/s (roughly the speed due to a 1m fall). Finally, we set the pulse time scale for O to be $h = \tau/5$. We find that this time scale provides sufficient resolution for (4.20) to accurately approximate the acceleration noise produced in typical rigid body collision scenarios.

Sound is evaluated using the time-series of collision impulses generated by a rigid-body solver. To avoid noise due to small impulses acting on bodies in static contact, we discard all impulses for which the relative collision velocity lies below some threshold v_{min}. In all of our examples, we used v_{min} = 5cm/s. We estimate a contact time scale τ for a rigid-body impulse occurring between bodies O_1 and O_2 at time t_0 with relative velocity exceeding v_{min}. As in §4.3.3, we make the simplifying assumption that the change in position/orientation of each body is negligible in the time period $[t_0, t_0 + \tau]$. We transform the listening position $\boldsymbol{x}$ in to each body's coordinate frame by applying the

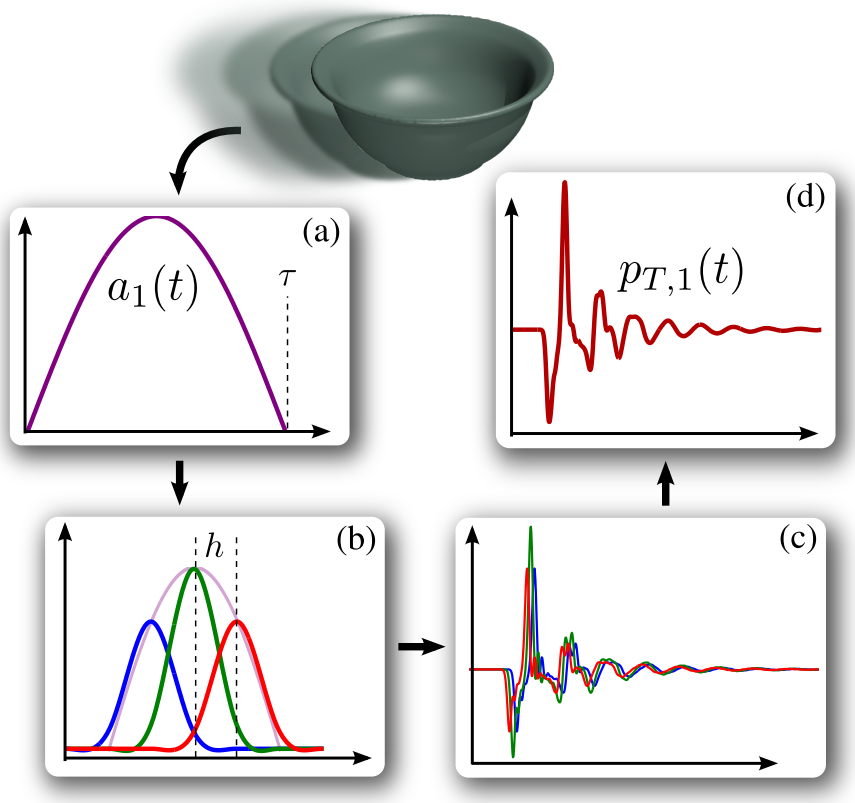


Figure 4.7: **Approximating impulse pressure:** *(a)* An impulse applied to the bowl results in acceleration along the bowl's x-axis with time-profile $S(t; t_0, \tau)$; *(b)* $S(t; t_0, \tau)$ is decomposed using appropriately scaled and offset instances of $\psi(t; h)$; *(c)* the corresponding directional pulse $p_{T,1}^{(h)}$ in the direction of listening position $\boldsymbol{x}$ is computed via (4.21), then scaled and offset according to the coefficients and offsets from part *(b)*; *(d)* the scaled and offset instances of $p_{T,1}^{(h)}$ are added together to reconstruct the full pressure signal experienced at listening position $\boldsymbol{x}$.

inverse rigid body transformations of O_1 and O_2 at time t_0. Finally, the pressure due to each body's acceleration is computed for these transformed positions using ((4.20)) with continuous acceleration signals determined by (4.9-4.10). This process is summarized in Figure 4.7.

4.5 Results

All sound and animation results can be found in the results video accompanying the paper associated with this work [21] [1].

Implementation Details: Rigid-body simulations are run with a solver based on [58]. Modal sound dynamics are driven using the same continuous forces used to produce acceleration noise (see §4.3). We integrate linear modal vibration dynamics using an implicit Newmark scheme [67] with a time step of $\Delta t = 1/192000$s and evaluate acoustic transfer due to modal vibration using the *FastBEM Acoustics* implementation (`www.fastbem.com`) of the fast multipole boundary element method [83, 119]. We found the popular Rayleigh damping model to be insufficient for generating realistic results in some of our examples. We extended this model with an additional damping term proportional the inverse stiffness matrix of the linear elastic system and found that the additional control provided by this model allowed us to produce much more realistic results for certain objects. All sounds were computed at a virtual microphone position of $\boldsymbol{x} = [1.0\ 1.8\ 1.0]^T$. For reference, the table appearing in our examples is centered at $x = z = 0$, elevated 40cm off the ground and has length 1.8m in the x-axis and width 0.8m in the z-axis. Modal sounds are delayed according to the distance between each object's center of mass and the microphone.

The precomputed solutions (4.24) are evaluated with a finite difference time-domain (FDTD) wave equation solver. We use the perfectly matched layers approach from [82] to solve the PDE on as small a domain as possible while avoiding reflections from the domain boundary. Equations (4.1-4.2) are time-stepped with a 2^{nd}-order leapfrog scheme using a 2^{nd}-order discretization in space via a staggered pressure/velocity grid (see [82] for details). We carefully compared results from our solver and FastBEM with

[1] `http://www.cs.cornell.edu/projects/sound/impact/`

analytical wave equation solutions in [75] to guarantee correct physical scaling between modal and acceleration sound.

Acceleration Noise Precomputation: Statistics for the acceleration pulse precomputation described in §4.4.2 are given in Table 4.1. In general, we find that objects with a mostly convex shape require fairly short precomputation simulations, while more concave objects tend to produce more acoustic intra-reflections and hence require longer simulation times.

Model	ρ (kg/m^3)	E (GPa)	ν	f_{min} (kHz)	Grid res.	T	h (μs)	Sim. time	Solve time	40×80 field size	5×10 field size
Ball Bearing	8940	123.4	0.34	159.5	125^3	1ms	7.3264	6m, 55s	2.7s	23.5 MB	0.42 MB
Bowl	2700	72	0.19	2.00	220^3	10ms	24.332	40m, 19s	2m, 16s	133 MB	2.15 MB
Coin	8940	123.4	0.34	10.07	215^3	2ms	4.8333	141m, 18s	3.3s	35 MB	0.6 MB
Dice	1200	2.4	0.37	32.9	170^3	1ms	14.898	17m, 38s	7.3s	35 MB	0.6 MB
Key	8940	123.4	0.34	2.03	350^3	1ms	5.56629	63m, 39s	46.4s	136 MB	2.3 MB
Key Ring	8940	123.4	0.34	3.94	245^3	1ms	5.56629	40m, 52s	5.0s	46 MB	0.8 MB
Mug	2700	72	0.19	1.32	195^3	13ms	25.467	115m, 59s	282m 1s	1131 MB	19 MB
Plate	2700	72	0.19	1.00	300^3	5ms	27.358	53m, 26s	28.1s	108 MB	1.8 MB
Rounded Dice	1200	2.4	0.37	29.2	170^3	1ms	17.469	16m, 58s	4.2s	35 MB	0.6 MB
Broken plate (11 pieces)	2700	72	0.19	Varied	Varied (4mm grid cells)	6ms	See §6.5	132m, 6s (all pieces)	3m, 17s (all pieces)	800 MB (all pieces)	14 MB (all pieces)
Broken glass pane (71 pieces)	2600	62	0.2	Varied	Varied (4mm grid cells)	2.5ms	See §6.5	13h, 12m (all pieces)	21m, 45s (all pieces)	6000 MB (all pieces)	95 MB (all pieces)

Table 4.1: **Model and Precomputation Statistics:** The finite difference grid resolution, simulation duration (T) and pulse time scale h used to precompute the pulse approximation introduced in §4.4.2. Simulation times and least-squares solve times are also provided. The field size columns indicate the memory usage to store the precomputed acceleration noise model at angular resolutions of 40×80 and 5×10. Timing/memory results for the fracture examples represent the time/memory taken to precompute/store the acceleration noise model for all pieces generated in the fracture simulation. Material parameters and the lowest modal vibration frequency (f_{min}) are provided for all objects. Simulations and solves were run on 8-core Intel Xeon X5570 and X7560 machines.

EXAMPLE (Ball bearing): Our simplest example is a spherical steel ball bearing with a radius of 7.5mm. All of the vibration modes for this object have frequencies well above the range of human hearing. As a result, modal sound synthesis produces no meaningful sound for this object. By including acceleration noise, we recover the familiar clicks produced when these objects collide.

EXAMPLE (Dice): We model dice with two different shapes. As in the case of the ball bearing, all vibration frequencies for these objects are inaudible. Our approach is able

to recover the familiar sound of dice rolling on a table.

EXAMPLE (Coin): For this example, we model a copper coin with roughly the same dimensions as an American quarter. While this object certainly produces significant ringing noise, the results produced by modal synthesis alone sound muffled and provide little temporal distinction between object impacts. The addition of acceleration noise allows us to recover detailed impact sounds.

EXAMPLE (Key and ring): As in the case of the coin, these objects have some audible ringing noise, but the inclusion of acceleration noise results in significantly more crisp, detailed sound.

EXAMPLE (Mug): Because the mug's shape, sustained acoustic reflections occur in its interior even for short boundary acceleration pulses. As a result, the acceleration noise for this object produces a noticeably pitched tone. To resolve this behavior we were required to run the precomputation simulation for a significantly longer time period than was sufficient for other examples. Due to the length of the signal associated with this simulation, the least-squares system (4.26) was very large, resulting in a costly TSVD solve.

EXAMPLE (Plate and bowl): These examples illustrate that as objects increase in size, the significance of acceleration noise relative to ringing noise diminishes. Nevertheless, the addition of acceleration noise in these examples still results in a subtle brightening of contact sounds.

EXAMPLE (Fracture): We use rigid-body fracture simulation and sound data generated by the methods of [137] to generate improved fracture sounds for animations of a breaking plate and a breaking pane of glass. Since fracture simulations tend to produce large ensembles of small objects, the absence of acceleration noise is particularly notice-

able. By including acceleration noise, we produce sounds with significantly enhanced temporal detail as objects break apart and the resulting pieces collide with each other. The plate example in particular illustrates that our model is also capable of resolving other continuous sound phenomena, such as rolling motion.

Precomputed Acceleration Noise Validation: Recall that when building the pulse approximations discussed in §4.4.2 the largest radius at which we sample pressure values is $4R$, where R is the bounding sphere radius of the object. We run a FDTD domain simulation of the wave equation using acceleration boundary conditions of the form (4.9-4.10) and sample the pressure time series at a set of listening locations located at a radius of $8R$ (well outside of the largest radial shell used for precomputation). We compare the directly simulated result to the pressure field computed using (4.20). We see that results computed with our method closely match ground truth FDTD result, with error generally manifested in the form of a small (on the order of a few samples) delay between the two signals (see Figure 4.8). We also note here that listening positions chosen for this evaluation were not chosen to be aligned with a direction from the discretized PAN field (see §4.4), indicating that our method does not suffer from large angular interpolation error.

Figure 4.8 also compares results computed using PAN models with different angular resolutions. Due to the smoothness of acceleration noise field, we see comparable results when using PAN fields discretized at resolutions of either 40×80 or 5×10. While this distinction does not influence synthesis time, it does affect memory usage (see Table 4.1). In Figure 4.9 we compare results from PAN fields with different pulse time scales h (see §4.4). All examples involving the bowl mesh are computed with a pulse time scales of $h = h_{base} = 24.322\mu$s. The following table provides synthesis times for the "multiple bowl drop" example synthesized with different time scales:

h	h_{base}	$2h_{base}$	$4h_{base}$	$8h_{base}$
Synthesis time (s)	3.15	1.98	1.25	0.89

Larger pulse time scales result in shorter synthesis times since fewer pulses need to be included for each impact. See the accompanying result video for a comparison of sounds synthesized using all four time scales. We observe little audible difference between sounds synthesized with pulse time scales of h_{base} and $2h_{base}$, suggesting that our method is effectively interpolating impact accelerations. However, synthesis with larger time scales results in audible differences and degraded sound quality, with many high-frequency features being omitted.

Precomputed Acceleration Noise Visualization: See the supplemental result video for animations of precomputed pressure fields (4.24) for some of our example objects.

Directional Resolution of Precomputed Solutions: We store precomputed acceleration noise pulses at two different angular resolutions and compare their sound contributions. While we certainly observe high numerical accuracy when using a finely discretized angular resolution of 40×80, the sounds produced by a much smaller field with an angular resolution of 5×10 are nearly indistinguishable. Table 4.1 lists the memory usage of these two fields for all objects.

We observe that we are able to obtain high accuracy using fields discretized at substantially lower resolutions than those used for far field acoustic transfer maps in Chapter 3. This is possible because precomputed acceleration noise fields tend to exhibit substantially less angular complexity than high-frequency modal transfer functions (compare, for example, Figure 4.4 with Figures 3.10 and 3.11).

Sound Synthesis Performance: Table 5.2 provides sound synthesis statistics for each of our examples. Synthesis times depends on the number of impulses produced by the

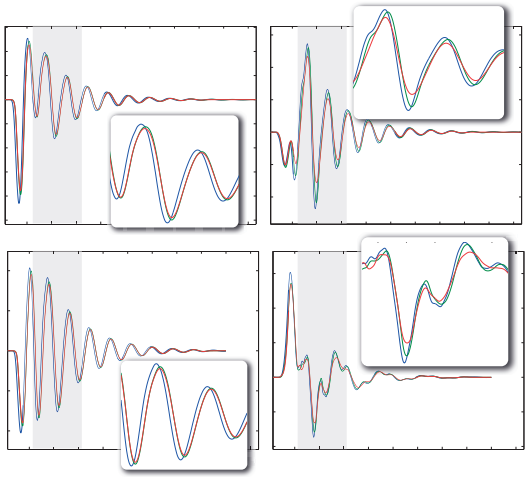


Figure 4.8: **PAN angular sampling comparison:** Pressure time series due to acceleration of a bowl mesh computed using an FDTD wave equation solver (blue), PAN discretized at angular resolution 40×80 (green) and PAN discretized at angular resolution 5×10 (red). We measure the pressure time series at two listening positions for both translational (top row) and rotational (bottom row) acceleration pulses with time profile $S(t; \tau/4, \tau)$ where $\tau = 0.0001$s. Inset figures magnify the highlighted regions of the original plots to make details more evident.

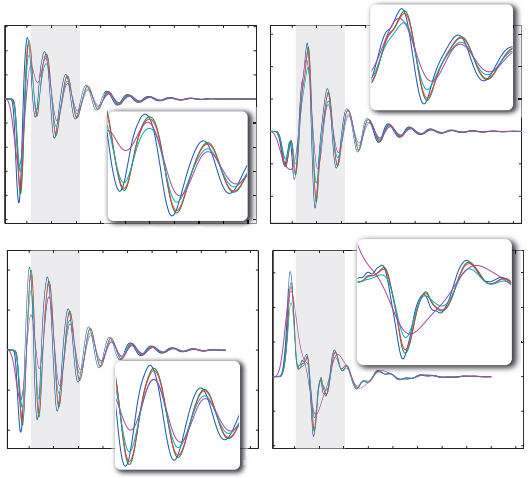


Figure 4.9: **PAN temporal resolution comparison:** Pressure time series due to acceleration of a bowl mesh. We compare results from a FDTD solver (blue), PAN solutions computed with pulse time scale $h = h_{base} = 24.322\mu$s (green), PAN solutions with $h = 2h_{base}$ (red), PAN with $h = 4h_{base}$ (cyan) and PAN with $h = 8h_{base}$ (magenta). We measure the pressure time series at two listening positions for both translational (top row) and rotational (bottom row) acceleration pulses with time profile $S(t; \tau/4, \tau)$ where $\tau = 0.0001$s.

rigid-body simulator, and the precomputed acceleration noise pulse lengths (§4.4.2) for each object.

Processed Results: For short transient sounds such as those produced by our method we find that the addition of environmental reverberation can produce more plausible results. We also apply dynamic range compression to certain results, since normalizing pressure time series to have unit ℓ_∞ norm tends to produce sounds in which some parts seem abnormally quiet. See the supplemental video for both dry and processed results for certain examples. Effects are added using *Adobe Soundbooth* during post-processing.

Example	Duration (s)	Δt (ms)	# impulses		Synthesis time (s)
Ball drop	5	2.5	1750916	(23411)	7.96
Coin drop	8	0.1	17307535	(22959)	16.00
Coin drop (w/ table)	8	0.1	3886215	(24045)	18.19
Dice drop (w/ plate)	5	1.0	26899	(3414)	0.88
Dice drop (w/ table)	5	1.0	44689	(683)	0.40
Key Chain	5	0.1	2052579	(48828)	12.36
Mug/Coin drop	5	1.0	383877	(3796)	2.68
Multiple bowl drop	5	1.0	23712	(1992)	3.29
Multiple mug drop	5	1.0	25147	(319)	10.33
Multiple plate drop	5	1.0	70007	(1303)	3.72
Plate fracture	5	0.025	3962503	(1293)	2.23
Glass fracture	2	0.025	11556813	(2507)	3.77

Table 4.2: **Sound Synthesis Statistics:** Acceleration sound synthesis times (averaged over 5 trials) for our examples. The duration and Δt columns report the length and time step duration for the rigid-body simulation. # impulses refers to number of impulses produced in the simulation (bracketed numbers are the number of impulses with relative velocity exceeding v_{min} – see §6.5). Synthesis was performed on an 8-core Intel X5570 machine.

4.6 Conclusion

We presented a practical model for synthesizing rigid-body acceleration noise. Sound is computed efficiently via precomputed approximations of the pressure field due to objects undergoing short acceleration pulses. This model allows us to recover an important component of rigid-body sound not modeled by traditional modal vibration. The addition of acceleration noise significantly improves the sound quality for a variety of rigid-body examples.

Limitations and Future Work: We make a number of simplifications when estimating a continuous contact force between colliding objects. Efficient and accurate determination of continuous contact forces is a challenging open problem. Effects such as inelasticity and variation due to contact region geometry may play a role in the production of acceleration noise. Moreover, our contact force model only considers normal contact.

As such, acceleration sounds due to frictional effects (e.g., a coin spinning on end, or a rolling ball) are difficult to synthesize [129]. Although computational methods exist to resolve these phenomena (e.g., [63]), they are computationally expensive and require careful parameter tuning. Further compression of the precomputed acceleration noise model (say, by individual compression of each directional time signal) and extending our model to real-time settings are other areas interesting areas for future work.

Our examples demonstrate that acceleration noise makes a significant contribution to rigid-body fracture sounds. For these examples we explicitly precompute solutions (§4.4) for each fragment produced by the fracture solver. In [137], the authors use proxy object geometry to avoid explicitly constructing modal sound data for each fracture object. This model could be expanded by introducing a similar proxy object model for acceleration noise.

As has been the case in previous work on rigid-body sound synthesis, we generate results independently for each object. This may fail to capture interesting and potentially important effects. Our experiments suggest that effects such as inter-object and environmental scattering can be particularly important for short, transient sounds such as acceleration noise or sound from highly damped modal vibration. Neglecting these effects may produce results that sound somewhat harsh or abrupt in some cases. For example, the acceleration noise produced by two small metal spheres colliding is altered dramatically by the addition of environmental reverb. Efficiently resolving both inter-object and environmental acoustic scattering for physically based animations such as the ones shown in our results is a challenging open problem.

CHAPTER 5

FASTER ACCELERATION NOISE FOR MULTIBODY ANIMATIONS USING PRECOMPUTED SOUNDBANKS

In this chapter, we build on the methods of Chapter 4 and introduce an efficient method for synthesizing rigid-body acceleration noise for complex multibody scenes. The synthesis algorithm described in the previous chapter requires object-specific precomputation. This is not a severe bottleneck for scenes like the ones presented in §4.5, since these animations involve only a few distinct objects. However, precomputing acceleration noise data for every object in a scene is prohibitively expensive for scenes involving rigid-body fracture or other sources of small, procedurally generated debris. The methods presented in this chapter avoid precomputation by introducing a proxy-based method for acceleration noise synthesis in which precomputed acceleration noise data is only generated for a small set of ellipsoidal proxies and stored in a *proxy soundbank*. Our proxy model is shown to be effective at approximating acceleration noise from scenes with lots of small debris (e.g., pieces produced by rigid-body fracture). This approach is not suitable for synthesizing acceleration noise from larger objects with complicated non-convex geometry; however, the results presented in §4.5 show that acceleration noise from objects such as these tends to be largely masked by modal vibration sound (consider, for example, the plate and bowl examples from §4.5). We manage the cost of our proxy soundbank with a new wavelet-based compression scheme for acceleration noise and use our model to significantly improve sound synthesis results for several multibody animations.

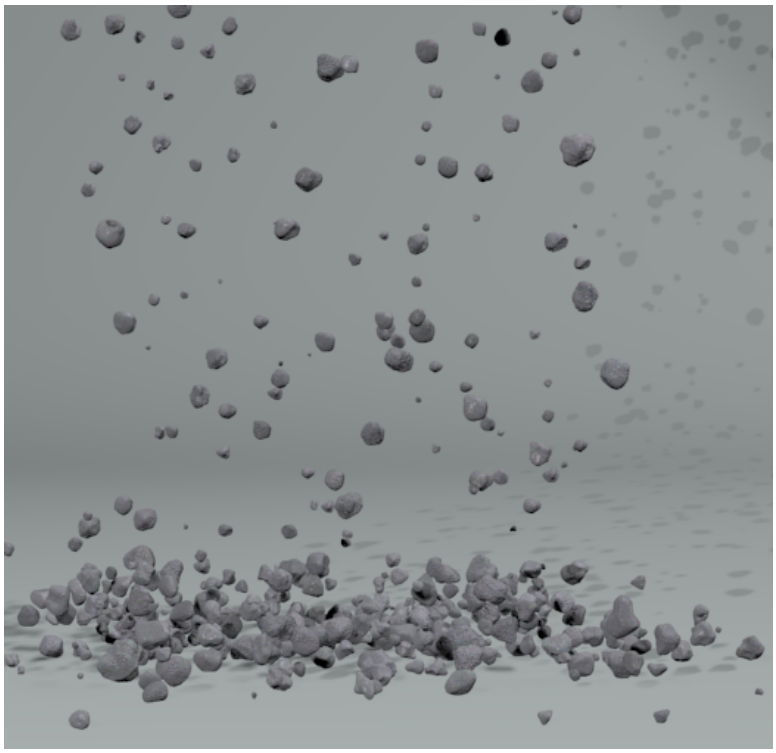


Figure 5.1: The acceleration sound synthesis algorithm from Chapter 4 requires per-object precomputation, making it impractical for scenes like this animation of 1000 falling rocks.

5.1 Introduction

Simulations of multibody dynamics for complex scenes – tumbling piles of rocks, shattering panes of glass, etc. – can be used to produce compelling animations with rich visual behavior. These simulations should also provide an equally rich source of sound. As we originally discussed in Chapter 4, the linear modal sound algorithm commonly used for animation sound synthesis fails to resolve acceleration noise. We showed in the previous chapter that acceleration noise is an important sound source for scenes involving small objects such as coins, dice, ball bearings, etc. The omission of acceleration noise is particularly noticeable when synthesizing sound from large ensembles of small objects (e.g., rigid debris generated from fracture simulations or some other procedural method).

The methods discussed in the previous chapter require extensive precomputation on a per-object basis. This approach guarantees physical accuracy and is not a severe bottleneck for scenes involving a few predetermined objects. However, it presents a challenge when attempting to synthesize sound from simulations with many unique objects. Moreover, the examples presented in §4.5 suggest that acceleration noise makes a large contribution relative to modal sound primarily in scenes with large ensembles of small objects. We also observe that the acceleration noise produced by such objects tends to be fairly simple when compared to sound produced by larger non-convex objects (bowls, mugs, etc.). As a result, it is important to develop simpler models for approximating sound from complex multibody scenes with many small objects (e.g., debris from fracture simulations). We address this with two main contributions:

1. *Proxy object soundbank*: We build PAN representations for a set of proxy ellipsoids and compute sound from arbitrary objects by fitting them to appropriate proxies based on their physical properties. By introducing a scaling relationship for the precomputed representation introduced in Chapter 4, we are able to limit the space of proxy ellipsoids to a few dozen objects.
2. *Memory-efficient PAN representation*: A common issue arising in both modal and acceleration sound synthesis is the large amount of storage required for precomputed sound data. To address this, we introduce a wavelet-based representation for precomputed acceleration noise. This enables acceleration sound synthesis which is more efficient in both time and space. Our wavelet-based PAN representation also requires a less costly preprocess than the representation used in Chapter 4. We further reduce the size of our proxy soundbank by fully exploiting symmetries arising in our precomputed data set, allowing us to limit the total size of this data set to approximately 5-26MB, depending on the amount of compression used.

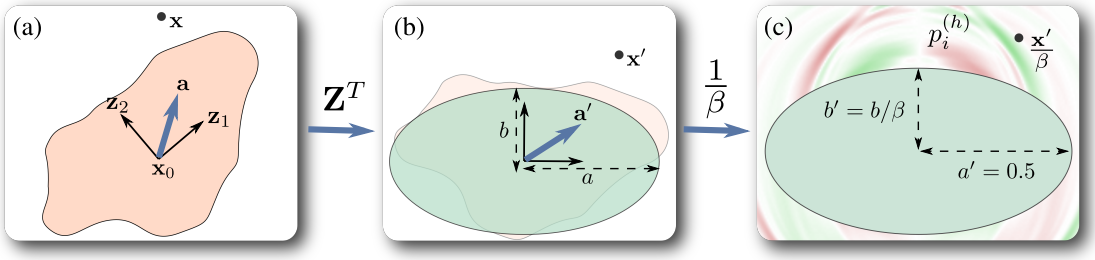


Figure 5.2: **Proxy Acceleration Sound Synthesis:** We synthesize sound due to motion of object O at listening position $\boldsymbol{x}$. *(a)* An object O undergoes translational acceleration $\mathbf{a}$. O's center of mass $\boldsymbol{x}_0$ and principle axes of inertia $\mathbf{z}_1, \mathbf{z}_2$ are shown; *(b)* We fit an ellipsoidal proxy to O according to its principle moments of inertia, and transform $\boldsymbol{x} \rightarrow \boldsymbol{x}', \mathbf{a} \rightarrow \mathbf{a}'$ in to the axis-aligned proxy ellipsoid space; *(c)* We scale the proxy ellipsoid to match a reference ellipsoid with unit x-axis length and synthesize sound using this ellipsoid's PAN functions $p_i^{(h)}$ and the scaling relationships (5.2) (in this figure, we assume $0 < \beta < 1$).

Other Related Work: In [137], the authors introduced a method for synthesizing modal sound from rigid-body fracture simulations. An important component in this method is an ellipsoidal proxy soundbank storing precomputed modal sound data. The results in §4.5 show that the addition of acceleration noise significantly improves the fracture sound results of [137] for certain scenes, but these results are limited to scenes with a small number of objects. In this chapter, we further improve upon these results by applying our proxy model to fracture simulations involving hundreds of pieces. Our work is similar in spirit to [137], but in this work we address the challenges of developing a proxy model for a fundamentally different sound phenomenon.

5.2 Acceleration Noise Proxy Geometry

In this section, we introduce an ellipsoidal proxy model for acceleration noise. By storing PAN fields for a small set of proxy objects, we are able to efficiently synthesize plausible acceleration noise for scenes with thousands of unique objects.

5.2.1 Scaling Relationships

In this section we establish scaling relationships between the acceleration noise produced by objects O and O_β, where O_β is identical to O but has been uniformly scaled by $\beta > 0$. Following the notation of §4.4.2, the boundary conditions used to solve the wave equation for pressure fields $p_i^{(h)}$ $(i = 1, \ldots, 6)$ due to object O are

$$\nabla p_i^{(h)} \cdot \mathbf{n}(\boldsymbol{x}) = \psi(t; h) \begin{cases} -\rho\, \mathbf{e}_i \cdot \mathbf{n}(\boldsymbol{x}) & i = 1, 2, 3 \\ -\rho\, (\mathbf{e}_{i-3} \times (\boldsymbol{x} - \boldsymbol{x}_0)) \cdot \mathbf{n}(\boldsymbol{x}) & i = 4, 5, 6 \end{cases} \tag{5.1}$$

where $\mathbf{e}_i \in \mathbb{R}^3$ is the vector with components $e_{ij} = \delta_{ij}$. Now, suppose that the exterior domains of O and O_β are Ω and Ω_β, respectively. Let P refer to pressure due to the scaled object O_β. The following scaling relationships hold:

$$P_i^{(\beta h)}(\boldsymbol{x}, t) = \begin{cases} \beta p_i^{(h)}\left(\frac{\boldsymbol{x}}{\beta}, \frac{t}{\beta}\right) & i = 1, 2, 3 \\ \beta^2 p_i^{(h)}\left(\frac{\boldsymbol{x}}{\beta}, \frac{t}{\beta}\right) & i = 4, 5, 6 \end{cases} \qquad \boldsymbol{x} \in \Omega_\beta. \tag{5.2}$$

See Appendix A for a proof of this result.

5.2.2 Proxy Soundbank

To avoid building PAN representations for each unique object in a scene, we instead map objects to ellipsoidal proxies according to their physical properties and build PAN representations for only these proxy objects. By exploiting the scaling relationships presented in §5.2.1, we can reduce the three dimensional set of all ellipsoids to a much smaller two dimensional set. Every ellipsoid in $\mathbb{R}^3$ is equivalent – up to scaling and rigid transformation – to an ellipsoid defined by $\frac{x^2}{A^2} + \frac{y^2}{B^2} + \frac{z^2}{C^2} = 1$ where $A = 0.5$; that is, ellipsoids with unit length in the x-axis. We also only need to consider ellipsoids for which $C \leq B \leq 0.5$. See §5.4 for soundbank precomputation details.

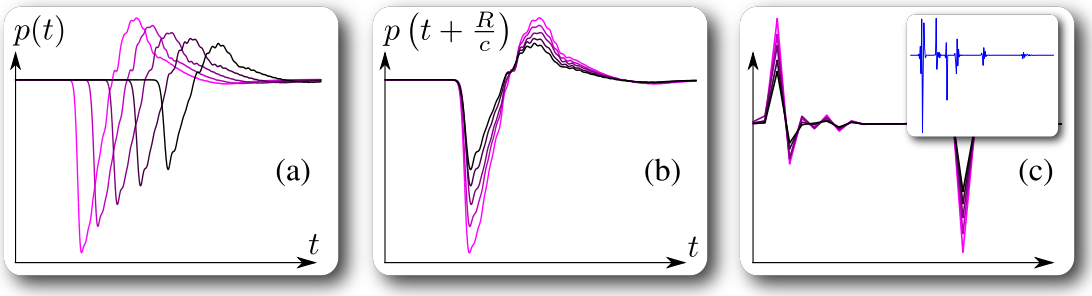


Figure 5.3: **Precomputed Acceleration Noise Compression:** *(a)* Acceleration noise signals evaluated with a wave equation solver at several radii R_i ($i = 1, \ldots, 5$) in a fixed listening direction; *(b)* Acceleration noise signals time-shifted according to (5.5); *(c)* A subset of the coefficients from wavelet decompositions of the time-shifted functions from *(b)*. The inset shows a larger set of wavelet coefficients for one of these functions. We compress PAN functions by storing only sufficiently large wavelet coefficients.

5.2.3 Proxy Sound Synthesis

Consider a rigid-body object O and let $\boldsymbol{x}_0$, $\mathbf{M}$ and V denote its center of mass, moment of inertia matrix, and volume. The ellipsoidal proxy used to represent O will be chosen so that its principle moments of inertia match those of O. A similar procedure was used to select proxy geometry for modal sound synthesis in [137]. $\mathbf{M}$ is real and symmetric and can be diagonalized to yield an orthonormal basis $\mathbf{Z}$ and diagonal matrix $\mathbf{D}$ such that $\mathbf{M} = \mathbf{Z}\mathbf{D}\mathbf{Z}^T$. The columns of $\mathbf{Z}$ and diagonal entries of $\mathbf{D}$ are the principle axes and principle moments of inertia for O [55]. For an axis-aligned ellipsoid with mass m, it is straightforward to derive $\mathbf{M}$ from the definition of the moment of inertia [55]:

$$\mathbf{M}_{ellipsoid} = \frac{m}{5}\begin{pmatrix} B^2+C^2 & 0 & 0 \\ 0 & A^2+C^2 & 0 \\ 0 & 0 & A^2+B^2 \end{pmatrix}. \tag{5.3}$$

Assuming that $D_{11} \leq D_{22} \leq D_{33}$, we set $\mathbf{M}_{ellipsoid} \equiv \mathbf{D}$ and solve the resulting system of equations to obtain ellipse parameters $A \geq B \geq C$. Next, we uniformly rescale (A, B, C) so that the volume of the resulting ellipsoid matches V. Assuming that this ellipsoid has the same density as O, identifying its volume with O's ensures that it also

has the same mass as O. The principle moments of inertia for this ellipsoid also match O's up to a scaling factor. This implies that O and its proxy ellipsoid exhibit similar rigid accelerations when subjected to the same external force and ensures that the magnitude of acceleration noise produced by this ellipsoid is consistent with the sound produced by O. Finally, we identify this ellipsoid with a proxy ellipsoid $(A' = 0.5, B', C')$ with unit length in the x-axis. We choose a scaling factor $\beta = 2A$ so that $(A, B, C) = (\beta A', \beta B', \beta C')$.

To synthesize sound from O, collision forces are estimated using O's original geometry and the methods discussed in §4.3. To approximate O's sound contribution at listening position $\boldsymbol{x}$, we begin by transforming $\boldsymbol{x}$ in to the coordinate frame of O's proxy ellipsoid: $\boldsymbol{x}' = \mathbf{Z}^T(\boldsymbol{x} - \boldsymbol{x}_0)$. Likewise, we rotate the translational and rotational accelerations $\mathbf{a}(t)$ and $\boldsymbol{\alpha}(t)$ applied to O to find the accelerations $\mathbf{a}'$ and $\boldsymbol{\alpha}'$ acting on the proxy ellipsoid: $\mathbf{a}'(t) = \mathbf{Z}^T\mathbf{a}(t)$, $\boldsymbol{\alpha}'(t) = \mathbf{Z}^T\boldsymbol{\alpha}(t)$. Suppose that O's proxy is parametrized by $(A, B, C) = (\beta A', \beta B', \beta C')$, where (A', B', C') has unit length $(A' = 0.5)$. Assuming that we have precomputed $p_i^{(h)}$ $(i = 1, \ldots, 6,\ h > 0)$ for O's unscaled proxy ellipsoid, we can evaluate $P_i^{(\beta h)}$ for the desired ellipsoid (A, B, C) using the scaling relationships (5.2). Finally, we use the PAN functions $P_i^{(\beta h)}$ and (4.20) to recover the total acceleration noise $P(\boldsymbol{x}', t)$ due to $\mathbf{a}'(t)$ and $\boldsymbol{\alpha}'(t)$ acting on the proxy ellipsoid (A, B, C). Figure 5.2 summarizes the process of fitting an ellipsoid to O and synthesizing sound from this proxy.

5.3 Proxy Soundbank Representation

The PAN representation introduced in §4.4.2 stores time signals q_k $(k = 1, \ldots, N)$ at a discrete set of angular directions surrounding an object. The signals in each direction are

explicitly discretized and stored at some sampling frequency f. Storing these fields at a reasonably high angular resolution can require on the order of 10-100 MB of storage per object. While this may not be particularly expensive in scenes with only a few unique objects, storing precomputed data for a large set of proxy objects could become prohibitively expensive if we use these methods directly. In this section, we discuss techniques for building a memory-efficient representation for our proxy soundbank.

5.3.1 Precomputed Acceleration Noise Compression

As we originally discussed in §4.4.2, precomputed acceleration noise functions $p_i^{(h)}$ are represented by discretizing the angular space (θ, ϕ) about object O's center of mass $\boldsymbol{x}_0$ and associating with each angular direction the series representation (4.21) for $p_i^{(h)}$. For the remainder of this section, we will drop subscripts and superscripts and refer to the function to be approximated simply as $p(\boldsymbol{x}, t)$. To find the values of the functions q_k in each direction (θ, ϕ) the true values of $p(R, \theta, \phi, t)$ are computed at a discrete set of radii $R_1, \ldots, R_M$ by solving (2.1) and (4.17) on O's exterior domain Ω. We can write $p(R_i, \theta, \phi, t)$ as a time series

$$\left\{p_i^0, p_i^1, p_i^2, \ldots\right\} \quad \text{where} \quad p_i^\ell = p(R_i, \theta, \phi, \ell\Delta t). \tag{5.4}$$

Δt is the simulation time step used to solve (2.1). In §4.4.2 and the associated results presented in §4.5, the functions $q_k(\theta, \phi, t)$ from (4.21) are also discretized at a sampling rate of $f = 1/\Delta t$. A least-squares system is built by enforcing the condition that (4.21) holds at each radius R_i and each time sample of (5.4). The system is solved for the full set of samples for the functions q_k in such a way that the functions are temporally smooth. This approach was successfully applied to a number of example objects; however, it tends to result in very large least-squares systems and memory-intensive representations for the resulting PAN fields.

We introduce a new fitting approach which simultaneously allows for compression of the PAN functions and a less expensive fitting process. Observe that if (4.21) holds with equality, then the time-shift $t - R/c$ in the right hand side can be moved to the left while preserving equality:

$$p\left(R, \theta, \phi, t + \frac{R}{c}\right) = \sum_{k=1}^{N} \frac{1}{R^k} q_k(\theta, \phi, t). \tag{5.5}$$

Figure 5.3 (a) illustrates the function $p(R, \theta, \phi, t)$ evaluated at five radii in a fixed listening direction and figure 5.3 (b) illustrates the time-shifted signals $p(R, \theta, \phi, t + R/c)$.

Rather than discretizing the PAN functions q_k and explicitly computing their samples, we instead choose to represent the functions q_k in a wavelet basis. Shifting $p(R, \theta, \phi, t)$ in time temporally aligns these signals, allowing us to represent them using the same wavelet basis. Figure 5.3 (c) illustrates some of the wavelet coefficients for the functions $p(R_i, \theta, \phi, t + R_i/c)$. We represent these signals using a Daubechies wavelet family [33]. Let $\widehat{\mathbf{p}}_i$ be the vector of coefficients in the wavelet basis for the function $p(R_i, \theta, \phi, t + R_i/c)$ and let $\widehat{p}_i^j$ be the j^{th} coefficient. Similarly, let $\widehat{q}_k^j$ be the j^{th} wavelet coefficient for the function $q_k(\theta, \phi, t)$ (θ and ϕ fixed). It follows from (5.5) and the linearity of the wavelet transform that

$$\widehat{p}_i^j = \sum_{k=1}^{N} \frac{1}{R^k} \widehat{q}_k^j. \tag{5.6}$$

(5.6) encodes a $M \times N$ least-squares system for each wavelet coefficient, where M is the number of sampling radii, and N is the number of series terms (5 and 2, respectively, in our examples). We solve (5.6) repeatedly to recover the full set of wavelet coefficients for the PAN functions q_k. These small, $M \times N$ least-squares systems are significantly easier to solve than the systems appearing in §4.4.2, which tended to have matrix dimensions numbering in the thousands (recall that these systems required a costly truncated SVD decomposition). As in §4.4.2, we discretize the angular space surrounding object

O, and repeat this procedure for each direction. This provides us with a representation of $p_i^{(h)}$ which can be efficiently evaluated at each point in space and time.

We compress the PAN fields q_k by only storing certain wavelet coefficients. From Figure 5.3 it is clear that many wavelet coefficients in $\widehat{\mathbf{p}}_i$ are close to zero. We define a tolerance $\epsilon > 0$ and choose to store coefficient j if and only if $\exists i : |\widehat{p}_i^j| \geq \epsilon \left\|\widehat{\mathbf{p}}_i\right\|_\infty$.

In practice, we use (4.20) to write the total pressure as

$$p(\boldsymbol{x}, t) = \sum_{k=0}^{\infty} \sum_{i=1}^{6} z_i(kh) p_i^{(h)}(\boldsymbol{x}, t - kh) \tag{5.7}$$

The inner sum can be evaluated efficiently by summing wavelet coefficients for each directional PAN field and performing a single wavelet reconstruction.

5.3.2 High-frequency Suppression

Suppose that the PAN fields discussed in §5.3.1 are stored at some sampling frequency f (so that $\Delta t = 1/f$ in (5.4)). To evaluate $p_i^{(h)}$, we must reconstruct the time signals q_k – sampled at frequency f – from their wavelet coefficients. Note, however, that when invoking the scaling relationships (5.2) the effective sampling frequency for $P_i^{(\beta h)}$ is $f' = f/\beta$. Assuming that f is relatively high (96 kHz for our proxy ellipsoids) and β is small, the effective frequency f' may significantly exceed frequencies necessary for high-quality audio synthesis (44-96 kHz). Fortunately, our wavelet PAN representation provides us with a convenient way to reconstruct the signals q_k at approximately the desired output sampling frequency, while suppressing content above this frequency.

As before, let $\widehat{\mathbf{q}}_k$ be the vector of wavelet coefficients for the time signal q_k, which is assumed to be sampled at frequency f. Coefficients are stored in $\widehat{\mathbf{q}}_k$ as follows: $\widehat{\mathbf{q}}_k(0)$ stores the wavelet smoothing coefficient, and $\widehat{\mathbf{q}}_k(2^\ell, \ldots, 2^{\ell+1} - 1)$ stores the detail

coefficients for level $\ell \geq 0$ of the wavelet basis. Intuitively, coefficients at higher indices in $\widehat{\mathbf{q}}_k$ represent higher-frequency content than coefficients at lower indices. Assuming that $\widehat{\mathbf{q}}_k$ has length T (assumed to be a power of 2), this storage scheme has the property that the vector $\widehat{\mathbf{q}}'_k = \widehat{\mathbf{q}}_k(0, \ldots, T/2-1)/\sqrt{2}$ stores the wavelet coefficients for a signal q'_k, which is sampled at frequency $f/2$, and is similar to q_k but lacks high-frequency details from the original signal. This relationship allows us to reconstruct the PAN time signal q_k with sampling frequency within a factor of 2 of the desired audio output frequency. Algorithm 2 summarizes the process of reconstructing scaled PAN functions q_k. Note, however, that in practice we do not independently reconstruct the functions q_k (see the last paragraph of §5.3.1). This process guarantees that we do not introduce aliasing artifacts by synthesizing details at frequencies significantly above the desired audio sampling frequency.

5.3.3 Ellipsoid Proxy Symmetries

We can further reduce PAN storage for ellipsoids by noting that ellipsoids centered at the origin are symmetrical about each axis. Moreover, symmetries in the boundary conditions (5.1) allow us to conclude that the following relationships hold for $x, y, z \geq$

0:

$$p_1(\boldsymbol{x}_0) = p_1(\boldsymbol{x}_1) = p_1(\boldsymbol{x}_4) = p_1(\boldsymbol{x}_5) \tag{5.8}$$
$$= -p_1(\boldsymbol{x}_2) = -p_1(\boldsymbol{x}_3) = -p_1(\boldsymbol{x}_6) = -p_1(\boldsymbol{x}_7)$$
$$p_2(\boldsymbol{x}_0) = p_2(\boldsymbol{x}_1) = p_2(\boldsymbol{x}_2) = p_2(\boldsymbol{x}_3) \tag{5.9}$$
$$= -p_2(\boldsymbol{x}_4) = -p_2(\boldsymbol{x}_5) = -p_2(\boldsymbol{x}_6) = -p_2(\boldsymbol{x}_7)$$
$$p_3(\boldsymbol{x}_0) = p_3(\boldsymbol{x}_2) = p_3(\boldsymbol{x}_4) = p_3(\boldsymbol{x}_6) \tag{5.10}$$
$$= -p_3(\boldsymbol{x}_1) = -p_3(\boldsymbol{x}_3) = -p_3(\boldsymbol{x}_5) = -p_3(\boldsymbol{x}_7)$$
$$p_4(\boldsymbol{x}_0) = p_4(\boldsymbol{x}_2) = p_4(\boldsymbol{x}_5) = p_4(\boldsymbol{x}_7) \tag{5.11}$$
$$= -p_4(\boldsymbol{x}_1) = -p_4(\boldsymbol{x}_3) = -p_4(\boldsymbol{x}_4) = -p_4(\boldsymbol{x}_6)$$
$$p_5(\boldsymbol{x}_0) = p_5(\boldsymbol{x}_3) = p_5(\boldsymbol{x}_4) = p_5(\boldsymbol{x}_7) \tag{5.12}$$
$$= -p_5(\boldsymbol{x}_1) = -p_5(\boldsymbol{x}_2) = -p_5(\boldsymbol{x}_5) = -p_5(\boldsymbol{x}_6)$$
$$p_6(\boldsymbol{x}_0) = p_6(\boldsymbol{x}_1) = p_6(\boldsymbol{x}_6) = p_6(\boldsymbol{x}_7) \tag{5.13}$$
$$= -p_6(\boldsymbol{x}_2) = -p_6(\boldsymbol{x}_3) = -p_6(\boldsymbol{x}_4) = -p_6(\boldsymbol{x}_5)$$

where $\boldsymbol{x}_0 = (x, y, z)$, $\boldsymbol{x}_1 = (x, y, -z)$, $\boldsymbol{x}_2 = (-x, y, z)$, $\boldsymbol{x}_3 = (-x, y, -z)$, $\boldsymbol{x}_4 = (x, -y, z)$, $\boldsymbol{x}_5 = (x, -y, -z)$, $\boldsymbol{x}_6 = (-x, -y - z)$, and $\boldsymbol{x}_7 = (-x, -y, -z)$. Our precomputed soundbank only stores PAN fields for directions in the positive $(x \geq 0, y \geq 0, z \geq 0)$ octant. We use (5.8-5.13) to evaluate these fields in all other octants.

5.4 Results

All sound and animation results can be found in the video accompanying the paper associated with this work [20]. [1].

[1] `http://www.cs.cornell.edu/projects/Sound/proxy/`

Algorithm 2: Reconstructs scaled precomputed acceleration noise functions at a sampling rate f' within a factor of 2 of the desired output sampling rate f_{out}. The `waverec` function reconstructs a time signal from T wavelet coefficients.

input : PAN wavelet coefficients $\widehat{\mathbf{q}}_k$, scaling factor β, PAN sampling rate f, output sampling rate f_{out}

output: Time signal $q'_k(t)$ and its sampling rate f'

begin
  $T \leftarrow \texttt{length}(\widehat{\mathbf{q}}_k)$
  while $f/\beta > 2f_{out}$ **do**
    $f \leftarrow f/2$
    $T \leftarrow T/2$
    $\widehat{\mathbf{q}}_k \leftarrow \widehat{\mathbf{q}}_k(0 : T-1)/\sqrt{2}$
  $q'_k(t) \leftarrow \texttt{waverec}(\widehat{\mathbf{q}}_k, T)$
  return $[q'_k(t), f/\beta]$
end

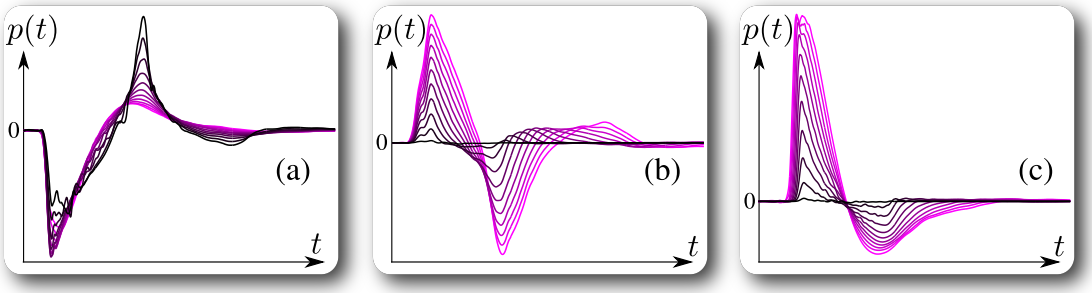


Figure 5.4: **Varying precomputed acceleration noise over the proxy soundbank:** The precomputed acceleration noise function $p_3^{(h)}$ (translation in the z-axis) evaluated with several proxy ellipsoids. *(a)* Varying ellipsoid parameter C between 0.025m and 0.5m while $A = B$ are held fixed at 0.5m. *(b)* Varying parameter B between 0.025m and 0.5m while $A = 0.5$m and $C = 0.025$m. *(c)* Varying parameters B and C simultaneously ($B = C$) between 0.025m and 0.5m with $A = 0.5$m.

Implementation Details: We synthesize sound using a precomputed soundbank with 66 ellipsoids. The ellipsoids have parameters $0.5\text{m} = A >= B >= C$ with B and C varying between 0.025m and 0.5m in increments of 0.0475m. We chose this increment to be sufficiently small to guarantee smooth variance of PAN fields across the proxy soundbank (see Figure 5.4). Since this set is well-sampled, we fit objects to the nearest ellipsoids in the soundbank rather than interpolating between ellipsoids, as the latter ap-

proach would require longer synthesis times. We evaluate the pressure time series (5.4) for each ellipsoid on a 500^3 finite difference grid with a time step of $\Delta t = 1/96000$s and use perfectly matched layers [82] to avoid reflections from the domain boundary. These high-resolution simulations were carried out over the course of several days on a set of eight 32-core Intel X7560 machines. Figure 5.6 illustrates the ellipsoids in our proxy set.

For these reference ellipsoids, we choose a time scale $h = 10^{-4}$s. This time scale was chosen based on results from §4.5. Specifically, the ball bearing example (a steel sphere of radius 0.0075m) in this section is assigned a time scale of $h_{ball} = 7.3 \times 10^{-6}$s. Our soundbank time scale of $h = 10^{-4}$ was chosen conservatively to guarantee that an equivalently scaled sphere from our proxy set will have a PAN time scale of approximately $h_{ball}/5$. We find that this time scale is sufficiently small to interpolate contact force profiles of the form (4.4) encountered in our simulations.

We compute the wavelet transforms discussed in §5.3.1 with a Daubechies wavelet family with 5 vanishing moments using the *GNU Scientific Library* implementation of the wavelet transform (`http://www.gnu.org/software/gsl/`). We find that this basis achieves a suitable compromise between performance and compression.

The positive octant $(x \geq 0, y \geq 0, z \geq 0)$ associated with each proxy is discretized by uniformly triangulating the unit sphere in this octant with 64 triangles and 45 vertices. Each vertex represents a direction in which PAN data is stored, and we use linear interpolation to synthesize sound in arbitrary directions. Our experiments show that storing proxy data at this resolution does not introduce significant errors relative to solutions computed with a finite difference solver.

Sound from multibody examples like the ones simulated for this chapter tend to ex-

hibit high dynamic range. As a result, normalizing pressure time series to have unit infinity norm tends to produce sounds in which certain parts are abnormally quiet. We address this by post-processing our results with dynamic range compression using *Adobe Soundbooth*. We also present some results post-processed with artificial environmental reverb.

Precomputed Acceleration Noise Compression: We find that with a PAN compression parameter of $\epsilon = 0.01$ produces compressed PAN fields that exhibit small errors relative to the explicit precomputed solutions (5.4) (on the order of 1-5%). While increasing this parameter does increase numerical error, noticeable differences in sounds synthesized using our proxy soundbank only become apparent at higher values of ϵ. See the accompanying result video for comparisons of sounds synthesized from soundbanks with varying ϵ. The following table details proxy data storage sizes for numerous values of ϵ (for reference, uncompressed PAN fields stored at the same resolution require 293MB):

ε	0.01	0.02	0.04	0.08	0.16	0.32	0.64
Size (MB)	26	20	16	12	7.3	4.5	2.6

The compressed PAN representation introduced in §5.3.1 is of general use, even for examples not computed using proxies. We apply the wavelet fitting procedure to several example objects from §4.5 and compare our results to the explicit PAN approach from Chapter 4. See Table 5.1 for a comparison of memory usage/synthesis times and the supplemental video for a comparison of acceleration noise results computed with these two approaches.

Proxy Validation: In §4.5, we compute acceleration noise for two fracture simulations by explicitly building PAN representations for every piece produced in the simulations. Using the same simulation data, we compare these results with sounds computed using

Model	PAN size (MB)		Synthesis time (s)	
	Explicit PAN	Current result	Explicit PAN	Current result
Plate	108	12	3.72	2.02
Mug	1131	76	10.33	3.42
Dice	35	6.9	0.40	0.19
Rounded Dice	35	6.1		
Coin	35	9.3	16.00	7.70

Table 5.1: **Precomputed Acceleration Noise Compression:** We compare memory use and acceleration sound synthesis times to those from §4.5 for a selection of models and example scenes. Results are reported for PAN fields with 3200 discrete angular directions to coincide with the original PAN results from §4.5. For all examples, we choose the wavelet compression tolerance to be $\epsilon = 0.04$. This was determined experimentally as roughly the largest ϵ we could use before producing noticeably different results.

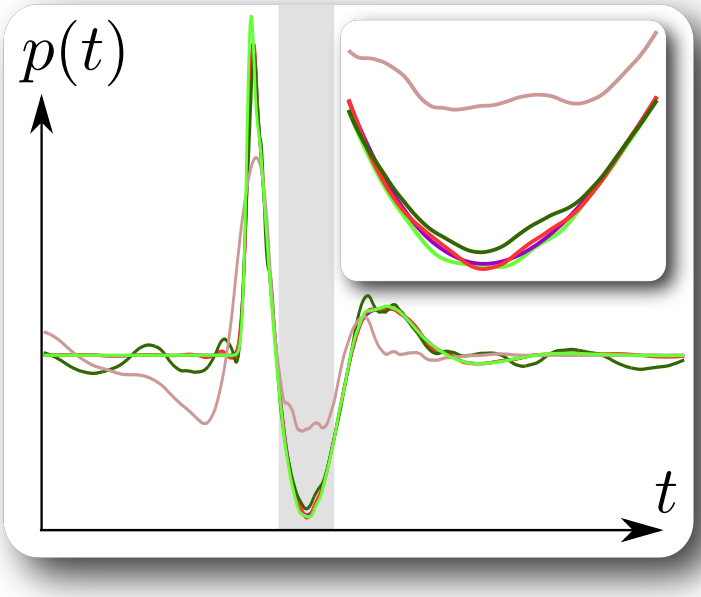


Figure 5.5: **Varying wavelet compression:** We visualize $p_5^{(h)}$ at a fixed position with varying levels of wavelet compression. The object considered here is an ellipsoid with $a = 0.5$m, $b = 0.405$m and $c = 2625$m. The inset shows a close-up of the highlighted region. Signals compressed with $\epsilon = 0.01$ and $\epsilon = 0.04$ (purple and red, respectively) exhibit good agreement with the finite difference solution (light green) with small errors arising from angular discretization. Fields compressed with $\epsilon = 0.16$ and $\epsilon = 0.64$ (dark green and pink, respectively) exhibit more significant errors.

our proxy soundbank. The original approach requires many hours of precomputation to build PAN fields for each object in these scenes. This approach also requires hundreds to thousands of MB of storage for PAN fields. Our method avoids this cost by synthesizing

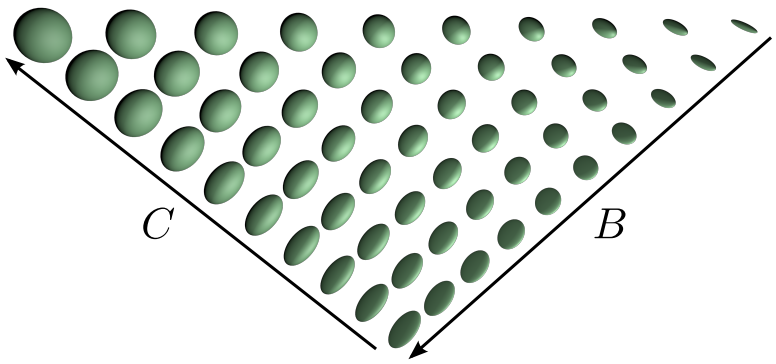


Figure 5.6: **Ellipsoid Proxy Soundbank:** All ellipsoid objects for which PAN fields are precomputed. Our results are computed by fitting objects to scaled ellipsoids from this set.

all acceleration noise with ellipsoidal proxies. We also present comparisons with sounds synthesized using a simpler proxy model in which each object O with volume V is approximated by a spherical proxy with volume V. We find that this method results in significant degradation of quality compared to our results. In particular, the contribution of acceleration noise tends to be severely underestimated by this method. This suggests that our approach is indeed capturing acceleration noise phenomena that is difficult to resolve with simpler techniques. See the supplemental video for these comparisons.

EXAMPLE (Rock Pile): To test the scalability of our method, we model a scene with 1000 unique, procedurally generated rocks and synthesize acceleration noise from the resulting simulation. We compute two examples: one with small rocks ($\approx$ 1-5cm in diameter) and one with larger rocks ($\approx$ 2-20cm in diameter). While the example with smaller rocks produces some modal sound, acceleration noise dominates this result. The example with larger rocks produces significantly louder modal sound, but the addition of acceleration noise still complements this example by introducing details not present in the modal result.

EXAMPLE (Glass Fracture): In this simulation, a glass pane falls to the ground and

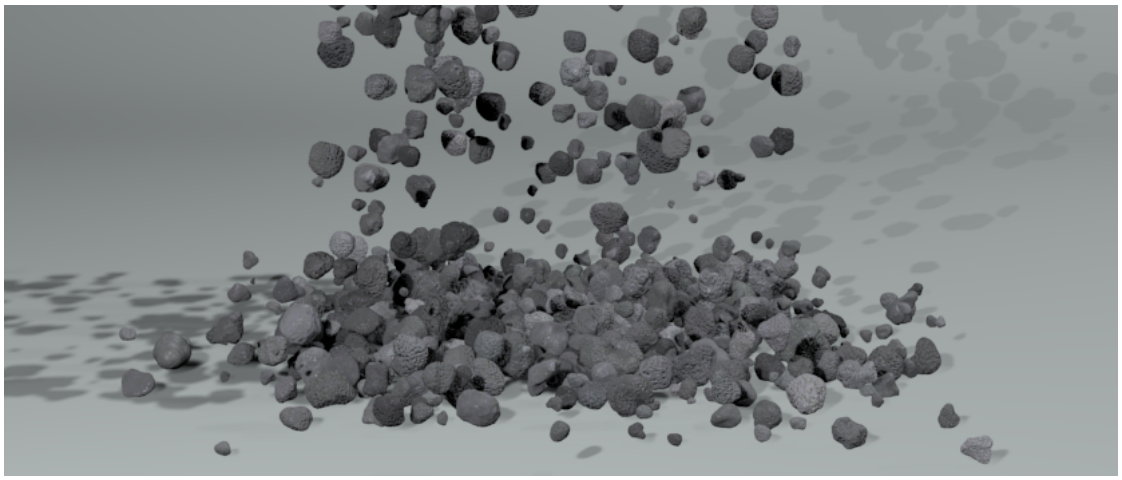


Figure 5.7: **Rock pile:** Synthesizing acceleration noise for this falling pile of 1000 procedurally generated rocks would require extensive precomputation to exactly resolve each object's contribution. Instead, we approximate each object with a proxy ellipsoid and synthesize acceleration noise with data from our precomputed soundbank.

shatters in to 315 small pieces. Without acceleration noise the debris produces very little sound.

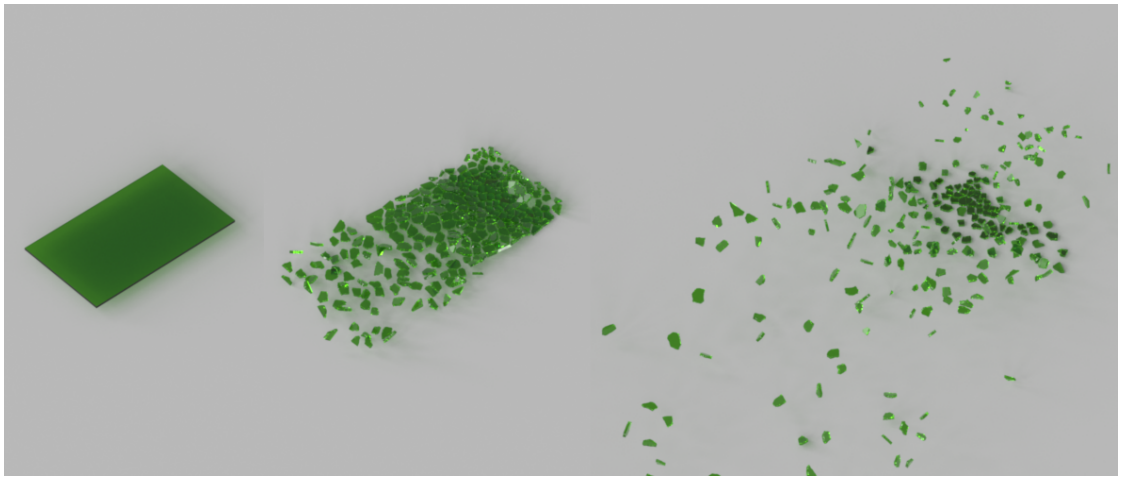


Figure 5.8: **Glass fracture:** This fracture simulation generates over 300 small objects with no audible vibration modes. Our method allows us to recover sound from this example by efficiently synthesizing acceleration noise for each piece using our proxy soundbank.

EXAMPLE (Breaking Plates): We simulate 10 plates falling to the ground and breaking. Many of the objects generated in this simulation produce no modal sound and the

Example	Duration (s)	Δt (ms)	# impulses	Synthesis time (s)
Single plate fracture	5	0.025	1293	1
Multiple plate fracture	4	0.25	5921	40
Glass fracture (71 pieces)	2	0.025	2507	2
Glass fracture (316 pieces)	3	0.025	18787	12
Rock pile (large)	6	0.25	210741	192
Rock pile (small)	6	0.25	96579	97

Table 5.2: **Sound Synthesis Statistics:** Acceleration sound synthesis times for our examples. The duration and Δt columns report the length and time step duration for the rigid-body simulation. # impulses refers to number of impulses used for sound synthesis.

addition of acceleration noise produces a substantially richer and more detailed result.

5.5 Conclusion

We presented an efficient method for synthesizing rigid-body acceleration noise from complex multibody scenes with hundreds to thousands of objects. We avoid precomputing acceleration noise data for each object in a scene by introducing an ellipsoid proxy model for acceleration sound. We build a soundbank of precomputed acceleration noise data for a set of ellipsoid proxies and limit the size of this data set by making use of a new wavelet compression scheme for precomputed acceleration noise data. As a result, the proxy soundbank only requires between 5 and 26MB of memory, depending on the amount of compression applied. This method introduces significant detail when applied to rigid-body fracture simulations and other simulations with large quantities of procedurally generated debris.

Limitations and Future Work: Our method computes sound independently from each object in a scene and adds these sounds together to recover the complete result. Ignoring acoustic interactions between objects may fail to capture interesting sound phenomena,

particularly in scenes involving many bodies stacked on top of each other (e.g., Figure 5.7). Existing brute force methods for resolving this phenomena are far too costly for animation sound synthesis. Developing efficient methods for resolving acoustic interactions between objects for both modal and acceleration sound is a challenging problem and an interesting area for future work.

Our results currently include only modal and acceleration sound from the objects in each scene. We do not currently synthesize sound from the ground plane. Zheng and James [137] synthesized modal sound for fracture examples, and included modal sound from the ground plane by synthesizing sound from a concrete slab. Including ground plane noise would likely enhance the realism of our results somewhat.

Our experiments show that our proxy-based synthesis pipeline is particularly effective for scenes involving small debris-like objects, producing results similar to those generated with object-specific precomputation. This is advantageous, as it is precisely objects like this for which acceleration noise is the dominant sound source. While our method does not accurately predict acceleration noise for large, non-convex objects, the contribution of acceleration noise for these objects is typically less significant relative to that of modal sound. Nevertheless, enriching our proxy database with additional object categories to better approximate acceleration noise from larger, non-convex objects is an interesting area for future work.

CHAPTER 6

ANIMATING FIRE WITH SOUND

The preceding chapters in this thesis have primarily addressed sound synthesis for rigid or nearly rigid objects. While this field had received some attention in earlier related work, the results produced by existing algorithms were frequently unsatisfactory. Chapters 3-5 introduce methods for producing rigid-body sound results with significant improvements over the results produced by earlier methods. However, it is certainly not the case that rigid body objects are the only important source of sound in virtual environments. Consequently, we now move away from the problem of rigid-body sound and address sound produced by a completely different phenomenon. We propose a method for synthesizing plausible fire sounds that are synchronized with physically based fire animations. To enable synthesis of combustion sounds without incurring the cost of time-stepping fluid simulations at audio rates, we decompose our synthesis procedure in to two components. First, a low-frequency flame sound is synthesized using a physically based combustion sound model driven with data from a visual flame simulation run at a relatively low temporal sampling rate. Second, we propose two bandwidth extension methods for synthesizing additional high-frequency flame sound content: (1) spectral bandwidth extension which synthesizes higher-frequency noise matching combustion sound spectra from theory and experiment; and (2) data-driven texture synthesis to synthesize high-frequency content based on input flame sound recordings. Various examples and comparisons are presented demonstrating plausible flame sounds, from small candle flames to large flame jets.

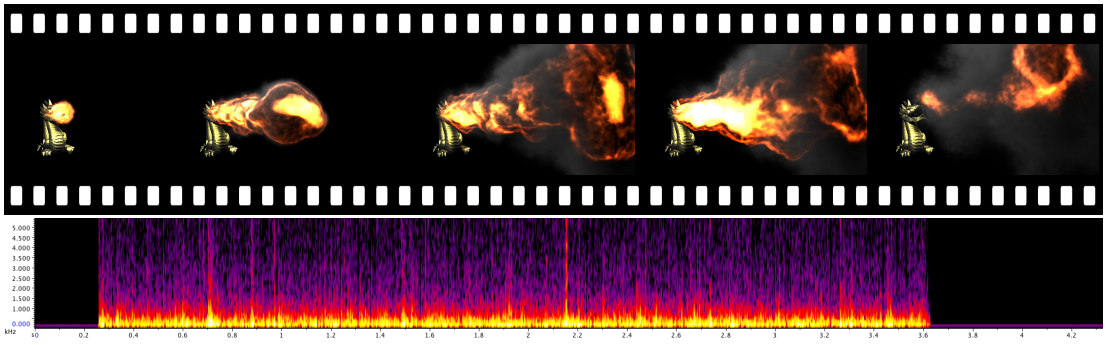


Figure 6.1: **Fire Sound Synthesis:** Our method produces the familiar sound of roaring flames synchronized with an underlying low-frequency physically based flame simulation. Additional mid- to high-frequency sound content is synthesized using methods based on spectral bandwidth extension, or sound texture synthesis for user-controlled flame sound styles.

6.1 Introduction

Candle flames, stove top burners and campfires are all familiar combustion phenomena. Larger flame sources such as flamethrowers, burning wreckage and fire-breathing dragons are familiar fixtures in the special effects industry. Due to the unsteady nature of combustion, these structures all tend to behave as noisy sound sources. Physically based fire simulators are capable of producing compelling visual simulations modeling all of these phenomena. Unfortunately, in spite of their ability to produce rich visual behavior, these solvers produce little information suitable for direct synthesis of flame sounds. Recorded combustion sounds can provide compelling auditory feedback, but they can require manual intervention, and can fail to produce realistic synchronized sounds which match visual flame behavior. While physically based sound synthesis methods have been developed for vibrating solid bodies [99, 100, 129], fracturing solids [137], aerodynamic phenomena [35, 36] and splashing fluids [136, 93], none exist for synthesizing the familiar sound of flames.

In this chapter, we present a hybrid method for synthesizing plausible sounds due to

combustion phenomena (see Figure 6.1 for a preview of our results). Rather than building a custom flame solver specifically for sound synthesis, we instead design a sound model which can be driven by data from current physically based animations. Using our sound model, existing simulators can synthesize synchronized sounds. However only low-frequency sounds, such as rumbling from very large flames, can be synthesized in practice for two reasons: (1) time-stepping combustion phenomena at audio rates is impractical due to the high computational costs of 3D flame simulation; and (2) real combustion noise results from complex thermo-acoustics of chemically reacting flows which are unresolved by most flame animations. Sounds recorded in high-speed video experiments (see §6.6) reveal detailed temporal behavior at a variety of time scales which cannot be resolved by flame solvers run at just graphics rates. With this in mind, we propose a hybrid technique in which flame sounds are physically simulated at only a few hundred Hertz, then additional mid- to high-frequency details are synthesized procedurally using one of two approaches:

1. *Spectral bandwidth extension* synthesizes high-frequency power-law noise to extend the frequency response of our otherwise low-frequency sounds. Frequency-domain sound synthesis methods are used to blend synchronized noise, and thereby produce power-law spectra that matches theoretical and experimental models of turbulent flames.

2. *Sound texture synthesis* allows us to synthesize fine-scale sound structure with more interesting temporal details. A modified coarse-to-fine texture synthesis method [133] is used to coherently synthesize detail on top of the input low-frequency physics-based sound. By varying the flame sounds used for input training data, users can control the style of synthesized sounds, while retaining synchronization with simulated flames. Recently, An et al. [121] synthesized sound for cloth animations using a hybrid method which also combines a simple physi-

cal model with data driven synthesis. Although similar in principle to our method, the technical approach used in their work is quite different from ours.

Other Related Work: The most closely related graphics work is the work on aerodynamic sound rendering by Dobashi et al. [35]. They devise a method for efficiently rendering aerodynamic sounds due to vortex-based noise [66], and effectively apply it to swinging clubs and sticks, and whistling wind. Their run-time efficiency is due to the use of precomputed flow noise signals. In a follow-up work [36], a method is proposed for general aerodynamic flow noise, and several examples are presented which involve sounds produced by turbulent flames. While these examples demonstrate the versatility of the aerodynamic sound model, we point out that the dominant source of combustion sound is not vortex-based noise [115]. For example, results from numerical and laboratory experiments confirm that another source of sound (namely, direct combustion sound; see §6.2) is the primary contributor to combustion noise, and that aerodynamic noise typically makes a relatively small contribution [69]. In engineering, numerical methods exist for combustion noise based on resolving the thermo-acoustics of chemically reacting flows [115], however the simulation methods (such as large eddy simulation) can be orders of magnitude more expensive than visual flame simulations, which would greatly limit the practicality of fire sound synthesis.

Non-physics-based synthesis of flame sounds synchronized with animation have also been considered, but can lack close synchronization with 3D flame state. Frequency-domain sound synthesis methods, descendant from spectral modeling synthesis [116], have been used to synthesize the noise-like roar, hiss and crackle of an animated fireplace [87]. Our noise-based bandwidth extension method uses similar frequency-domain noise methods to enhance our low-frequency physics-based sound. It was inspired by (blind) bandwidth extension methods used in the audio community to add

high-frequency detail to degraded signals such as digitally encoded audio [81, 79, 4]. McDermott et al. [89] synthesized a fire sound clip using spectral noise, and found that missing temporal structure could be modeled using higher-order marginal statistics. In contrast, we use a spectral noise model and obtain temporal structure from a low-frequency physics-based fire sound.

Sound texture modeling and synthesis provides another way to re-synthesize fire sounds from recordings (see [124] for a recent review), and includes audio generalizations of image texture synthesis [38, 133, 39]. Granular synthesis [111] is a classic method for re-synthesizing micro-sound details, but it can be difficult to arrange grains so as to re-synthesize meso-scale temporal structures. Wavelet tree learning [37] is able to re-synthesize stochastic and quasi-periodic textures effectively, but offers no automatic input control needed for fire sound synthesis. Motion-driven sound synthesis [17] provides a fully automatic control technique wherein a training motion signal w/ sound is segmented and used to map sound onto an input motion signal, e.g., of a 2D car motion. Unfortunately, it does not generalize to 3D fire motions, and our experience with synthesizing short segments of audio (using low-frequency sound segments to index high-frequency content) suffered from obvious granular-synthesis-like artifacts. In contrast, we use a modified texture synthesis approach wherein low-frequency fire sounds seed a coarse-to-fine sound texture synthesis similar to [133].

6.2 Background

6.2.1 Physically Based Flame Simulation

Our sound synthesis approach is designed to build upon existing flame solvers familiar to the computer graphics community [98, 64, 65]. In particular, our examples are generated using the *Pyro FX* solver and *Flame Solver* from *Side Effects Software*'s *Houdini* 3D animation tools package (`http://www.sidefx.com`). The flow of gaseous fuel and products in a 3D domain is modeled using the Euler equations

$$0 = \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} + \frac{1}{\rho} \nabla p \quad \text{where} \quad \nabla \cdot \mathbf{u} = \phi. \tag{6.1}$$

Here ϕ is an optional *divergence source* term introduced in the vicinity of combustion to model the effect of expanding gases [43]. A density field is used to govern the introduction, advection and diffusion of smoke [42]. A temperature field is also modeled to track the introduction of heat by combustion and to drive forces such as thermal buoyancy [48]. Other effects such as vorticity confinement and procedural turbulence may be introduced to increase the liveliness and visual realism in simulated flames [98, 65].

Combustion is modeled in [98] using a *blue core* model. In premixed flames (flames in which reactants are mixed and will burn immediately upon reaching a certain temperature) the blue core specifies a *flame front* interface between unburned fuel, and hot gaseous products produced by rapid combustion as fuel crosses the flame front. Separate sets of incompressible flow equations are used to model the flow of unburned fuel and gaseous products inside and outside of the flame front. The level set method is used to track a moving implicit surface representing the flame front. Voxels outside of the flame front store reaction coordinates which track the time elapsed since the gas stored in each voxel crossed the front. These quantities are combined with a flame temperature

profile function to track the temperature of gaseous products outside of the front.

In Houdini's Pyro FX solver, fuel is stored as a volume field along with temperature, velocity, etc. Fuel has an associated ignition temperature T_I and burn rate b. At a given time step, if voxel (i, j, k) has fuel concentration $f_{ijk} > 0$ and temperature $T_{ijk} > T_I$ then a quantity of fuel $\Delta f_{ijk} = \Delta t b$ is consumed and removed from voxel (i, j, k). Here Δt refers to the simulation time step. Temperature T_{ijk}, divergence sourcing ϕ_{ijk} and density ρ_{ijk} are modified according to how much fuel is consumed. To make the flames more lively, synthetic turbulence can be injected in to the velocity field [14, 74].

6.2.2 Combustion Sound Generation

In general, sounds produced by combustion phenomena can be expressed as a sum of contributions from multiple sources. A wave equation for combustion sound can be derived which includes contributions from turbulent vortex-based flow noise and direct combustion noise [27]. Numerical and experimental results in [69] suggest that direct combustion noise is indeed the dominant source of sound from combustion phenomena. The primary quantity of interest in our work is direct combustion noise; that is, noise produced by unsteady density fluctuations resulting from combustion heat release. This noise source is modeled using the inhomogeneous wave equation (2.3) presented in §2.1.1. Specifically, the following equation describes direct combustion sound: [27, 106]:

$$\frac{1}{c_0^2}\frac{\partial^2}{\partial t^2}p - \nabla^2 p = -\frac{\partial}{\partial t}\left(\frac{\rho_0\left(\gamma - 1\right)q}{\rho c^2}\right) \tag{6.2}$$

p, c, ρ and γ refer to the unsteady pressure, speed of sound, density and ratio of specific heats of air, respectively; c_0 and ρ_0 refer to the ambient speed of sound and density; γ is assumed to be independent of temperature, and combustion is assumed to take place at

ambient pressure p_0. The heat release rate q is the rate at which heat is introduced in to the domain by combustion. As we originally discussed in §2.1.1, the free-space Green's function (2.5) can be used to solve (6.2) for the sound pressure [27],

$$p(\boldsymbol{x}, t) = \frac{1}{4\pi} \frac{\gamma - 1}{c_0^2} \frac{\partial}{\partial t} \int_{\mathbb{R}^3} \frac{[q]}{\|\boldsymbol{x} - \mathbf{y}\|} d^3\mathbf{y}, \tag{6.3}$$

where brackets $[\cdot]$ refer to evaluation at $t - \|\boldsymbol{x} - \mathbf{y}\| / c_0$.

6.2.3 Combustion Sound Spectra

Numerous theoretical and experimental investigations in to the acoustic power spectrum emitted by flames have been carried out. Abugov and Obrezkov [1] recorded sounds generated by flames in a propane burner, and observed that in the high frequency region (beyond a certain peak frequency) the power spectrum of acoustic emissions from the flames exhibits a $P(f) \propto f^{-5/2}$ power law. A theoretical study of combustion sound by Clavin and Siggia [30] showed that the $f^{-5/2}$ power law would result if the flame front were subjected to fully developed velocity turbulence that obeyed a Kolmogorov energy spectrum, $E(k) \propto k^{-5/3}$ where k is the wave number. Note that Kolmogorov turbulence models are commonly used in fluid animation [74]. More recent experimental work studying the acoustic emissions of turbulent flames [106] has also suggested that beyond a certain peak frequency the power spectrum of combustion sounds appears to obey a power law $P(f) \propto f^{-\alpha}$, and fitting this model to experimental data suggested plausible values for α in the range $2.1 < \alpha < 3.4$.

6.3 Low-Frequency Fire Sound Synthesis

6.3.1 Fire Sound Model

Under the premixed flame assumption, unburned gas is consumed and releases heat via combustion very quickly as it crosses the flame front [98]. Strahle [123] argued that only velocity fluctuations in the vicinity of the flame front are responsible for significant sound output. To model low-frequency sound output from a flame simulation, we make the assumption that the heat release due to velocity fluctuations in the vicinity of a flame front surface patch δS about point $\boldsymbol{x}$ is proportional to the velocity flux across that patch

$$\delta q(\boldsymbol{x}) = \mathbf{u}(\boldsymbol{x}) \cdot \mathbf{n}(\boldsymbol{x})\ \delta S \tag{6.4}$$

where $\mathbf{n}(\boldsymbol{x})$ is the normal to the surface patch δS at point $\boldsymbol{x}$. Using this, the volumetric heat release integral can be rewritten as an integral over the front surface S

$$\int_{\mathbb{R}^3} q\ d^3\boldsymbol{x} = \int_S \mathbf{u} \cdot \mathbf{n}\ dS. \tag{6.5}$$

Intuitively, this model suggests that the rate at which heat is released in to the domain at a point $\boldsymbol{x}$ on the front surface is proportional to the rate at which fuel is carried in to the surface at that point. Relationships between the volume integral of the heat release rate and surface velocity flux integrals of this form occur in the literature [123, 30, 27]. We ignore time delays and distance attenuation in (6.3), and also omit constants, since they just provide a fixed scaling to the output sound. Time delays and attenuation could, in principle, be included in our model, but are unlikely to dramatically alter our results. Using these simplifications and (6.5), we can generate low-frequency flame sounds using

$$p(t) = \frac{d}{dt} \int_{S(t)} \mathbf{u}(\boldsymbol{x}, t) \cdot \mathbf{n}(\boldsymbol{x}, t)\ dS(\boldsymbol{x}) = \frac{d}{dt} I(t) \tag{6.6}$$

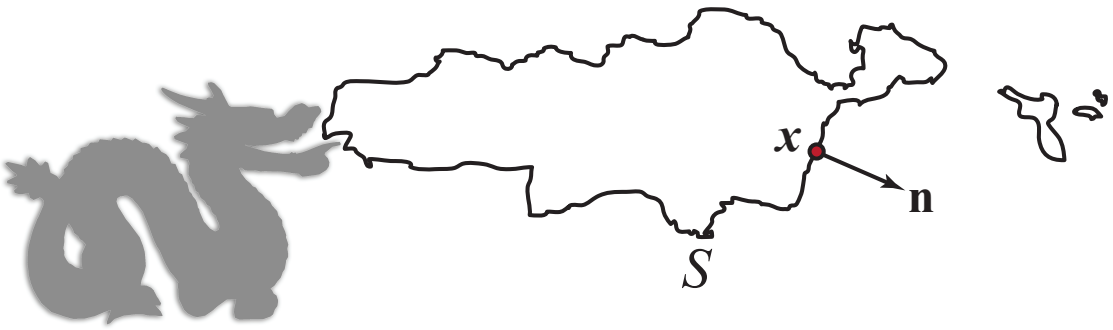


Figure 6.2: **Flame front surface:** Sound is computed by evaluating and differentiating a velocity flux integral $I(t)$ over this dynamic surface.

where the flame front S, its normal $\mathbf{n}$ and the velocity field $\mathbf{u}$ are all time-dependent quantities. Figure 6.2 provides a schematic of the geometry used in this sound model.

6.3.2 Flame Front Estimation

Since flame solvers model the flow of smoke, gaseous products, fuel, etc., the velocity field $\mathbf{u}$ is readily available at any point in the domain (see §6.2.1). In some solvers, a representation of the flame front may also be available. For example, level set methods track an implicit surface representing the flame front [98], from which an iso-surface can be extracted using Marching Cubes [85] or a similar method.

However, as described in §6.2.1, not all flame solvers use an explicit representation of the flame front to model combustion. In these cases, we must estimate a representation of the flame front surface which is physically plausible and resolves the turbulent dynamics. Given a fuel-based combustion model, such as the one presented in §6.2.1, we can define a field B_{ijk} whose contents specify the current rate of fuel consumption in voxel (i, j, k). For example, this information is stored in the "burn" field in Houdini's solver. By extracting a suitable iso-surface, we can estimate a plausible interface separating unburned fuel and burned combustion products.

Extracting an iso-surface directly from data on a modest-resolution voxel grid tends to produce jagged surfaces with non-smooth normal fields. Unfortunately such surfacing artifacts can result in undesirable sound artifacts with our method. To address these problems we use a higher-order cubic interpolation method to reconstruct S. Suppose that we wish to build an iso-surface from field B, which is discretized on a finite difference grid at points B_{ijk}, which corresponds to the value of B at point (x_i, y_j, z_k). In order to recover a smooth front surface S and normal field $\mathbf{n}$ we introduce interpolation functions $\phi_{ijk}(\boldsymbol{x})$ and define a new field $\tilde{B}$ as follows:

$$\tilde{B}(\boldsymbol{x}) = \sum_{i,j,k} \phi_{ijk}(\boldsymbol{x}) B_{ijk} \quad \text{where} \quad \sum_{i,j,k} \phi_{ijk}(\boldsymbol{x}) = 1. \tag{6.7}$$

We define $\phi_{ijk}(\boldsymbol{x})$ as a product of one-dimensional interpolation functions $\phi_{ijk}(\boldsymbol{x}) = \phi(\frac{x-x_i}{h})\phi(\frac{y-y_j}{h})\phi(\frac{z-z_k}{h})$, where h is the voxel grid size, and ϕ is a cubic B-spline basis function,

$$\phi(t) = \frac{1}{6} \begin{cases} -3\tau^3 + 3\tau^2 + 3\tau + 1, & |t| \leq 1 \\ (1+\tau)^3, & 1 \leq |t| \leq 2 \\ 0, & \text{otherwise,} \end{cases} \tag{6.8}$$

where $\tau \equiv 1 - |t|$. Extracting S and $\mathbf{n}$ from $\tilde{B}(\boldsymbol{x})$ defined in this manner provides a smoother iso-surface and normal field (see Figure 6.3), and greatly reduces sound artifacts.

6.3.3 Sound Pressure Evaluation

The simulations with which we are trying to synthesize sound are generally only run with time-stepping rates of hundreds of steps per second. To evaluate a pressure signal at audio sampling rates we interpolate the original simulation-rate signal. The pressure

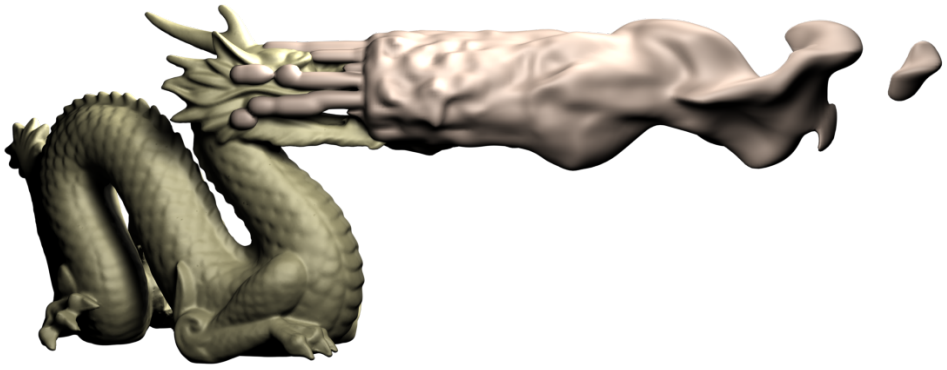


Figure 6.3: **Flame front surface** estimated using the method in §6.3.2.

signal used to generate sound has the form

$$p(t) = \frac{d}{dt} \int_{S(t)} \mathbf{u}(\boldsymbol{x}, t) \cdot \mathbf{n}(\boldsymbol{x}, t) \, dS(\boldsymbol{x}) = \frac{dI}{dt}. \tag{6.9}$$

Since $I(t)$ is known only at discrete time steps $t_0, t_1, t_2, \ldots, t_N = 0, \Delta t, 2\Delta t, \ldots, N\Delta t$, we use interpolation functions $\psi_i(t)$ to form its interpolated signal,

$$\tilde{I}(t) = \sum_{i=0}^{N} \psi_i(t) I(t_i) \quad \text{where} \quad \sum_{i=0}^{N} \psi_i(t) = 1. \tag{6.10}$$

The sound pressure $p(t)$ can then be evaluated at any point in time,

$$p(t) = \frac{d}{dt} \sum_{i=0}^{N} \psi_i(t) I(t_i) = \sum_{i=0}^{N} \psi_i'(t) I(t_i). \tag{6.11}$$

We use a Mitchell-Netravali cubic filter (4.19) for our interpolation function $\psi_i(t)$.

This sound synthesis method can generate low-frequency rumbling noises characteristic of combustion (see §6.6). Unfortunately, it is unsuitable for mid-to-high frequency sound synthesis, such as a flickering candle flame sound. The large cost associated with fluid solvers makes running fire simulations at audio rates impractical. Furthermore, more sophisticated combustion chemistry models are needed to properly resolve the complex heat release rates. As such, it is unlikely that running expensive, audio-rate simulations with graphics fire solvers would yield desirable results. We therefore investigate hybrid sound synthesis techniques in §6.4 and §6.5.

6.4 Spectral Bandwidth Extension

In this section, we present a bandwidth extension method for adding physically plausible power-law noise to enhance low-frequency fire sounds. Using time-frequency processing we modify our sounds' frequency spectrum so that, beyond a certain frequency, it matches a preferred fire spectra [139, 87]. Recall from §6.2.3 that experimental and theoretical results suggest that in the mid- to high-frequency range, flame sounds exhibit a power-law spectrum $f^{-\alpha}$, with $\alpha \in [2.2, 3.4]$.

Background (spectral noise synthesis): Given a power spectrum $f^{-\alpha}$, the corresponding frequency-domain pressure amplitudes are $|\tilde{p}(f)| = f^{-\alpha/2}$. To synthesize time-domain noise, we first assign uniformly random phases in the range $\theta_j = [-\pi, \pi]$ to each frequency-domain sample j to obtain a frequency-domain sound pressure, $\tilde{p}_j(f) = f_j^{-\alpha/2} e^{-i\theta_j}$. Next, time-domain noise with the desired spectrum is recovered by taking the inverse Fourier transform, $N(t) = IFFT(\tilde{p})$. In practice, to form $N(t)$ we first apply a high pass filter F_{high} to $\tilde{p}$ to suppress frequencies below a cutoff f_{cut}. This ensures that the main content of $N(t)$ is indeed high-frequency noise rather than low-frequency rumbling.

Spectral noise extension: In order to appropriately synchronize power-law noise with simulated low frequency content, we choose a cutoff frequency f_{cut} at which we wish to begin replacing simulated low-frequency content with synthesized noise. In all of our examples we use the Nyquist frequency of our simulation's 360 Hz time-stepping rate: $f_{cut} = 180$ Hz. Next, we divide our low-frequency pressure signal in to windows of half-width w_H, where w_H is an integer number of samples. Algorithm 3 describes a time-frequency processing method to introduce high-frequency content in to each win-

Algorithm 3: Extends a low-frequency sound with power-law noise

input : pressure, w_H, α, G_{blur}
output: pressureExtended

begin
  pressureExtended $\leftarrow 0$
  for $i \leftarrow 0, 1, 2, \ldots$ **do**
    $w_i(n) \leftarrow \begin{cases} w_H - |n - iw_H| & \text{if } |n - iw_H| \leq w_H \\ 0 & \text{otherwise} \end{cases}$
    $[p_{W,i}, N_{W,i}] \leftarrow$ `WindowSignal` (pressure,w_i,α)
    $\beta \leftarrow$ `ChooseBeta` ($p_{W,i}$, $N_{W,i}$, α, G_{blur})
    $\tilde{p}_{W,blend} \leftarrow$ `Blend` (`FFT` ($p_{W,i}$), β `FFT` ($N_{W,i}$))
    $p_{W,blend} \leftarrow$ `IFFT` ($\tilde{p}_{W,blend}$)
    pressureExtended $\leftarrow$ pressureExtended + $p_{W,blend}$
  return pressureExtended
end

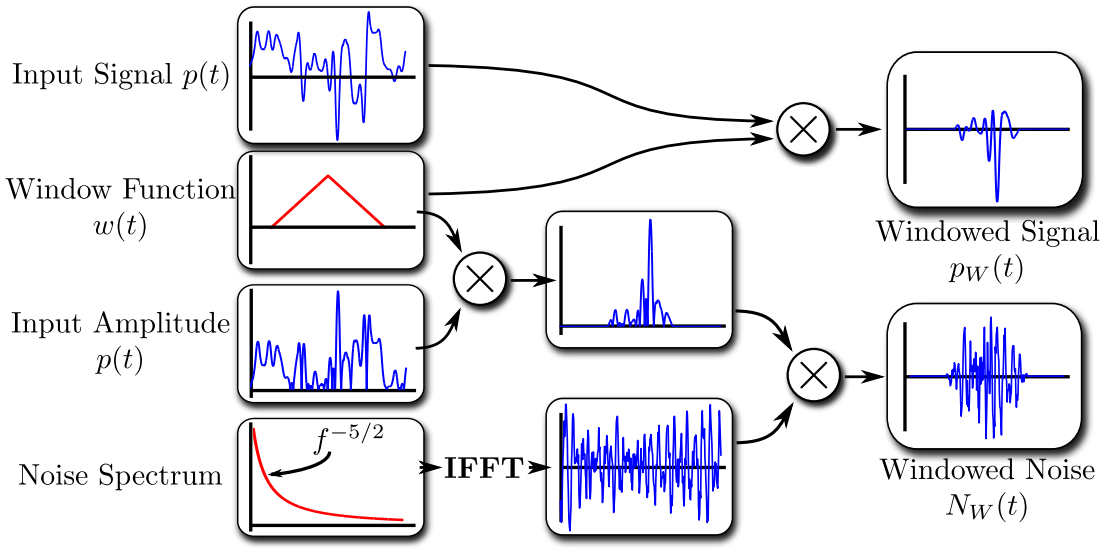


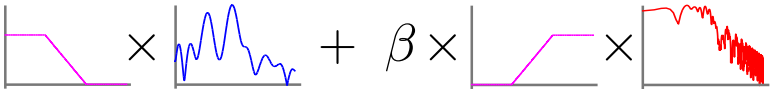
Figure 6.4: **WindowSignal function from Algorithm 3:** Given the input signal $p(t)$, we construct a windowed version $p_W(t)$ by multiplying with a linear window function $w(t)$. A noise signal $N(t)$ is generated by the inverse Fourier transform of a power law spectrum, and scaled by $|p(t)w(t)|$ to produce a windowed noise signal $N_W(t)$.

dowed signal. The windowed input signal $p_W(t)$ and a spectral noise function $N_W(t)$ are computed as shown in Figure 6.4; we scale the noise by the amplitude of the windowed pressure signal in order to synchronize its temporal structure. We wish to blend together $p_W(t)$ and $N_W(t)$ such that we preserve the original behavior of the pressure

spectrum in the neighborhood of f_{cut} and also so that the resulting pressure spectrum does not have any harsh discontinuities. This is done by choosing an appropriate scaling factor β for $N_W(t)$. We compute the spectra for each signal, $\tilde{p}_W(f) = \mathbf{FFT}(p_W(t))$ and $\tilde{N}_W(f) = \mathbf{FFT}(N_W(t))$, and linearly blend the two spectra (using high/low pass filters in some range surrounding f_{cut}),

$$\tilde{p}_{W,blend}(f;\beta) = F_{low}(f)\ \tilde{p}_W(f) + \beta\ F_{high}(f)\ \tilde{N}_W(f), \tag{6.12}$$

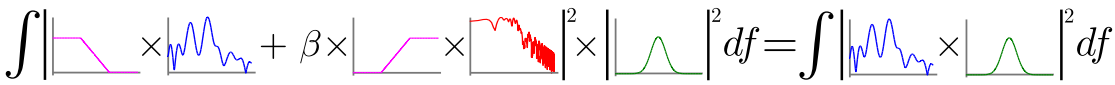
which can be illustrated as follows:



We estimate β by requiring that the integrated power of the blended signal $\tilde{p}_{W,blend}$ match that of the original input signal $\tilde{p}_W$,

$$\int |\tilde{p}_{W,blend}(f;\beta)\ g(f)|^2\ df = \int |\tilde{p}_W(f)\ g(f)|^2\ df, \tag{6.13}$$

where $g(f)$ is a weighting function used to ensure that signal power is only considered in a blend region near f_{cut}. We illustrate (6.13) as



We solve (6.13), a quadratic equation, for $\beta > 0$. In our implementation, $g(f)$ is a Gaussian with mean f_{cut} and standard deviation σ_{cut}. Finally, we use nearly the same parameters for all of our bandwidth-extension examples (see Table 6.1).

α	f_{cut}	σ_{cut}	w_H	Blend range
2.5–3.5	180.0 Hz	10 Hz	500 samples (11.34 ms)	$[165.0\text{Hz}, 195.0\text{Hz}]$

Table 6.1: **Parameters for Spectral Bandwidth Extension:** Parameters used for all spectral bandwidth extension results. Blend range refers to the linear-blending range used by $F_{low}(f)$ and $F_{high}(f)$.

6.5 Synchronized Sound Texture Synthesis

The bandwidth extension algorithm presented in §6.4 adds richness to sounds synthesized with the methods of §6.3 by adding synchronized and appropriately scaled noise with a physically plausible power spectrum. However, recorded sounds and high-speed flame videos indicate that additional meso-scale temporal structure exists. This structure could be added by synchronizing detailed recordings of fire sound with simulated flames, say, by scaling sound volume according to flame size. However, this simple approach would lack appropriate synchronization, e.g., a large, but steady flame makes very little noise. In this section, we propose a sound texture synthesis algorithm for synthesizing synchronized fire-sound details.

6.5.1 Synthesizing Fire-Sound Details

We synthesize fire-sound details in a coarse-to-fine manner using a modified texture synthesis algorithm most similar to [133]. Inputs to our algorithm are a fire sound signal p_S synthesized using the methods of §6.3, and a training audio clip p_T, e.g., a recording of real flame sounds. Our audio-detail-transfer method synthesizes a sound clip with the low-frequency temporal structure of p_S, but the fiery details of p_T.

Analogous to [133], we construct 1D Gaussian pyramids [16] G_T and G_S from our training and low-frequency input data, p_T and p_S. We refer to G_T and G_S as the

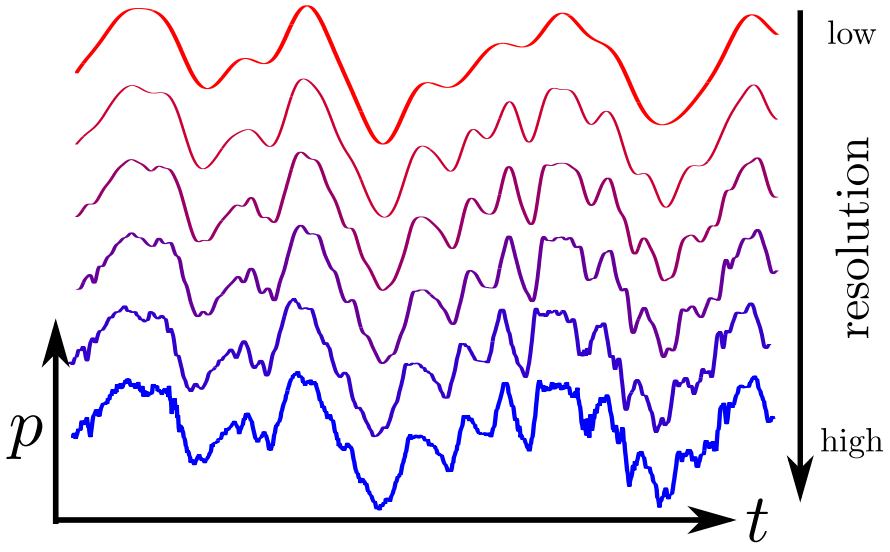


Figure 6.5: **Fire sound Gaussian pyramid** for a 23ms training sound p_T. All fire-sound details are synthesized in a coarse-to-fine manner using an $L=6$ level pyramid as shown here.

"training" and "signal" pyramids, respectively (Figure 6.5 illustrates G_T for a short fire sound). Let $G_T(\ell)$ and $G_S(\ell)$ refer to the samples stored at level $\ell = 1 \ldots L$, where L is the coarsest level. If $N_{T,\ell}$ (similarly, $N_{S,\ell}$) is the number of samples in level ℓ of the training (signal) pyramid, then we write the sequence of samples in the training (signal) pyramid as

$$p^0_{T,\ell}, \ldots, p^{N_{T,\ell}-1}_{T,\ell} \quad \left(p^0_{S,\ell}, \ldots, p^{N_{S,\ell}-1}_{S,\ell}\right). \tag{6.14}$$

A key distinction between our approach and that of Wei and Levoy [133] is that we build G_S from our input fire-sound signal p_S, rather than initializing it with random noise. We zero the contents of levels $1, \ldots, L-1$, only retaining p_S information in $G_S(L)$. Retaining the low-frequency signal enables us to synthesize fire-sound details synchronized with the low-frequency sound p_S. We synthesize levels of the signal pyramid from coarse to fine; synthesis starts at level $L-1$, since $G_S(L)$ is specified by the input data p_S.

6.5.2 Windowed Hierarchical Synthesis

Similar to Wei and Levoy [133], we synthesize the next "pixel" by finding an approximate nearest neighbor in a dictionary of multi-level neighborhood samples. However, for reasons of temporal coherence and efficiency, we choose to synthesize short, overlapping audio windows instead of individual samples. Consider windows on some level ℓ with window centers, $c^i = ih,\ i \in \mathbb{N}$; let the window half-width (in samples) be $h \in \mathbb{N}$ (we use $h = 4$) so that window i has sample indices in the range $\Omega^i \equiv [c^i - h, c^i + h]$. We synthesize level ℓ of G_S window-by-window in order of increasing window index, i. Suppose that $G_S(\ell)_{i-1}$ is a partially synthesized signal in which windows $0, 1, \ldots, i-1$ have been computed. We form $G_S(\ell)_i$ by adding signal p_{new} to the index range Ω^i, scaled by a linear hat function $W_i[n] = h - |n - c^i|$.

p_{new} is chosen from a set of training windows produced from p_T. Each level ℓ of G_T is divided in to windows with index ranges Ω^i. Window i is associated with a feature-value pair $\left(\mathbf{f}^i_{T,\ell}, \mathbf{p}^i_{T,\ell}\right)$ where $\mathbf{p}^i_{T,\ell}$ is the sub-vector of $G_T(\ell)$ with sample indices Ω^i. $\mathbf{f}^i_{T,\ell}$ is a two-level *feature vector* built from samples in $G_T(\ell)$ occurring immediately before window i in the range

$$\left[c^i_\ell - hh_f, c^i_\ell - h\right], \tag{6.15}$$

and neighboring samples from $G_T(\ell + 1)$ in the range

$$\left[(i/2)\, h - hh_f, (i/2)\, h + hh_f\right], \tag{6.16}$$

where h_f is a window half-width multiplier (we use $h_f = 3$, resulting in features with 46 dimensions). In practice, these samples do not lie at integral indices, in which case we use linear interpolation to determine their values. See figure 6.6 for an illustration of the structure of a feature vector. We build a dictionary $\mathbb{D}_\ell$ of feature-value pairs for each level $\ell = 1, \ldots, L-1$ (level L is omitted because it is initialized with our low-frequency fire sound). To synthesize window i of level ℓ we build a feature vector

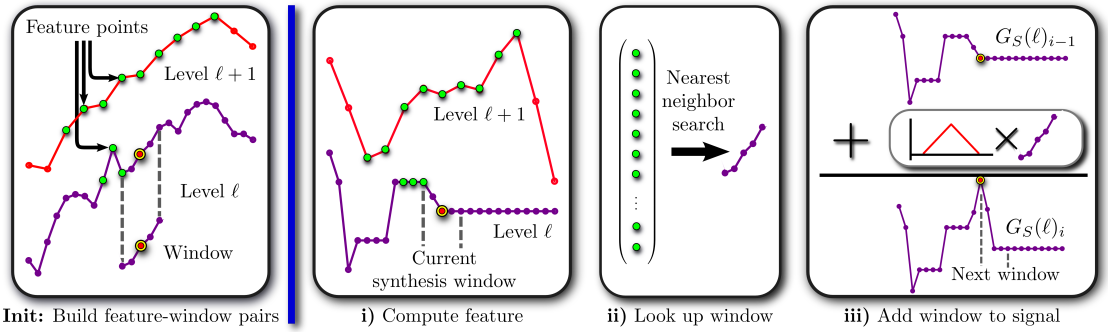


Figure 6.6: **Sound texture synthesis** ($h = 2, h_F = 2$)**:** Initialize the synthesis algorithm by building dictionaries $\mathbb{D}_\ell$ of feature-window pairs for levels $\ell = 1, \ldots, L-1$ of the training pyramid. Then, for levels $\ell = L-1, \ldots, 1$ of the signal pyramid; for window $i = 0, 1, 2, \ldots$ in level ℓ; **(i)** Build the window's feature vector, **(ii)** Look up the nearest neighbor in the training dictionary $\mathbb{D}_\ell$, **(iii)** add the resulting window (scaled by a linear hat function) to the signal at level ℓ and proceed to the next window.

of samples from $G_S(\ell)$ and $G_S(\ell+1)$ indexed by (6.15) and (6.16). p_{new} is chosen by performing a nearest neighbor search in the feature vector space of $\mathbb{D}_\ell$. Figure 6.6 illustrates the process of building $\mathbb{D}_\ell$ and synthesizing $G_S(\ell)$ for $h = 2, h_F = 2$ (we use $h = 4, h_F = 3$ in practice).

To improve efficiency, we search for approximate rather than exact nearest neighbors. Given a feature $\mathbf{f}^i_{S,\ell}$ whose nearest neighbor is distance d_{opt} away, we find a feature whose distance d from $\mathbf{f}^i_{S,\ell}$ is no more than $(1+\epsilon)\, d_{opt}$. In all examples, we use $\epsilon = 1$. Larger values will improve performance at the cost of sound quality. Approximate nearest neighbor queries are implemented using the *ANN* library (`http://www.cs.umd.edu/~mount/ANN`).

6.5.3 Dynamic Range Matching

The input and training signals p_S and p_T may have vastly different dynamic ranges. This problem must be addressed for the synthesis algorithm in §6.5.2 to produce meaningful

output. For instance, if p_S is much louder than p_T then there may be little temporal coherence from window-to-window in the nearest neighbors sampled from the dictionary $\mathbb{D}_\ell$, resulting in an sound clip with harsh temporal discontinuities. While manual normalization of p_S and p_T can help, an automated solution to this problem is desirable.

We use a histogram-matching procedure similar to the one proposed in [61]. We construct cumulative distribution functions F_S and F_T for the amplitudes of the lowest-resolution signals, $G_S(L)$ and $G_T(L)$, respectively. When synthesizing window i from level $L-1$ of G_S the feature vector $\mathbf{f}^i_{S,L-1}$ includes samples from level L in the range given by (6.16). We compute the average magnitude of these samples, $p_{abs,avg}$, then evaluate

$$p_{abs,training} = F_T^{-1}\left(F_S\left(p_{abs,avg}\right)\right). \tag{6.17}$$

Finally, when constructing $\mathbf{f}^i_{S,L-1}$ we scale all samples from $G_S(L)$ by $r = p_{abs,training}/p_{abs,avg}$. Intuitively, if exactly 90% of the signal data has magnitude below $p_{abs,avg}$ then r is chosen so that 90% of the training data lies below $r\ p_{abs,avg}$. Note that this procedure only takes place when choosing feature vector entries from level L of the signal pyramid.

6.6 Results

We now present sound synthesis experiments for a variety of combustion scenarios. All sound and animation results can be found in the video accompanying the paper associated with this work [19]. [1].

Implementation Details: All flame simulations were performed using Houdini's Flame solver (based on the *blue core* model described in 6.2.1) or Houdini's Pyro FX solver

[1] `http://www.cs.cornell.edu/projects/Sound/fire/`

(based on the alternate combustion model described in 6.2.1) and run at a time-stepping rate of 360Hz. Typical simulation times ranged between 2 and 10 hours depending on the length and resolution of the animation and the particular machine used. Iso-surface extraction is carried out using a custom Houdini surface operator (SOP) implemented using the *Houdini Development Kit* (HDK). Surfacing took between 5-20s per time step on a single core. We exploit sample-level parallelism and evaluate surfaces as a post-process in parallel on a cluster. Velocity flux integrals are computed on triangles of S using a one-point quadrature rule. All scenes were rendered using Side Effects Software's *Mantra* renderer. Training audio clips were selected from the *Ultimate Fire* sound library [2]. For texture sound synthesis, all training and input signals p_T and p_S had a 44.1 kHz sampling rate. Prior to either bandwidth extension or sound texture synthesis, we pre-process sound outputs from the synthesis method in §6.3 using a high-pass filter (with 30Hz cutoff frequency) to suppress low-frequency fluctuations which are inaudible but tend to produce loud, undesirable content when extended via §6.4 or §6.5.

Method Comparison: We provide comparisons between three sound synthesis approaches:

1. **Low-frequency sound synthesis:** Sounds produced using the velocity flux-based sound model presented in §6.3.
2. **Noise-based bandwidth extension:** Sounds produced with the bandwidth extension algorithm from §6.4. We note that although these sounds have a physically plausible power spectrum, they tend to lack interesting temporal behavior beyond what is predicted by the low-frequency synthesis method. Moreover, since all sounds are synthesized using the same noise model, sounds produced using this method tend to be very uniform across a variety of combustion scenarios.

[2] `http://www.therecordist.com/soundbox-sfx/soundbox-pro/ultimate-fire`

Example	p_T length (s)	p_S length (s)	$\lvert\mathbb{D}_1\rvert$	Synthesis time (s)	
				BE	STS
Dragon	15	9	167694	223	56
Candle	10	6	112070	151	86
Torch	16	5	179808	63	41
Flame Jet	15	10	167694	256	54
Burning Brick	8	5	98310	64	20

Table 6.2: **Synthesis statistics** for bandwidth extension (BE) and sound texture synthesis (STS) examples. $\lvert\mathbb{D}_1\rvert$ is the number of features at the highest resolution level of the training pyramid G_T. BE timings are for an unoptimized Matlab implementation.

3. **Sound texture synthesis:** Sounds generated using the texture synthesis approach detailed in §6.5. By varying the choice of training data used in this process, we find that we can produce a richer variety of sounds than those produced by pure noise-based bandwidth extension. Table 6.2 provides synthesis timings and statistics.

EXAMPLE (Dragon): We model a fire-breathing dragon roughly 1.2m in height. Fuel inflow from the dragon's mouth is varied to achieve dynamic variation in the sound.

EXAMPLE (Candle): This scene models a small candle-like flame blown by a turbulent wind force (see Figure 6.7 (Left)). Spectral bandwidth extension produces plausible results for this example due to the lack of complex mid-to-high frequency structure in small flames such as this.

EXAMPLE (Torch): A burning torch is animated to swing through the domain (see Figure 6.7 (Middle)) using Houdini's flame solver, which makes use of a blue core model [98] with detonation shock dynamics [64].

EXAMPLE (Flame Jet): Fuel is injected with high velocity in to a closed 1.5m wide box (see Figure 6.7 (Right)). Fuel inflow is animated to turn on and off periodically. As

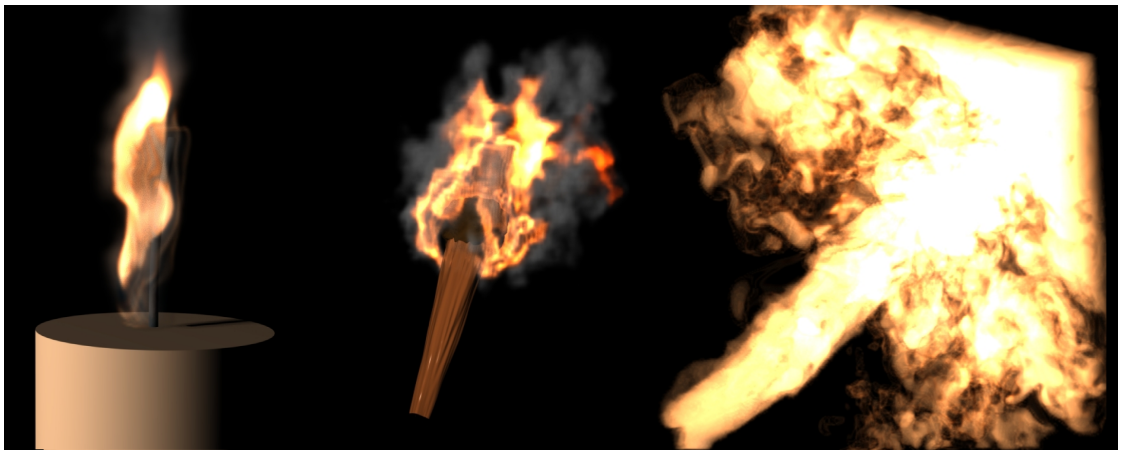


Figure 6.7: **Candle, Torch & Flame Jet Examples**

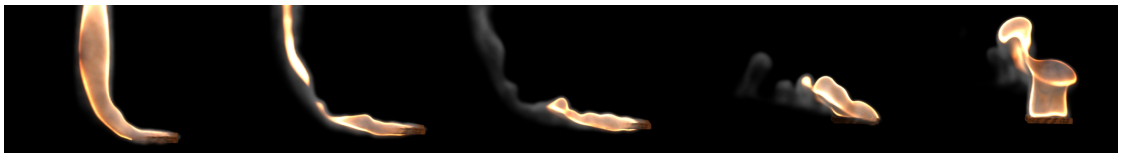


Figure 6.8: **Burning brick example** produces characteristic ruffling flame sounds as it moves side to side.

the fuel is shut off, we hear a "burn-off" effect as the remaining fuel is consumed.

EXAMPLE (Burning Brick): We model a small rectangular emitter animated to move rapidly through the domain (see Figure 6.8). As the source is moved, we hear characteristic ruffling noises.

COMPARISON (Simulated vs. recorded sound): Sounds and high-speed video (600 FPS) of a small "fire starter" brick were recorded by the authors. The "Burning Brick" example is modeled to resemble this experiment so that comparisons can be made. Results are also generated in which the recorded sound itself is used as training data for the texture synthesis method from §6.5.

COMPARISON (Varying power-law exponent, α): We provide a series of results for the "Burning Brick" simulation synthesized using spectral bandwidth extension with varying power law exponents α. α is treated as a parameter that can be tuned to change

the character of the output sound. $\alpha = 3.0$ seems to produce reasonable results, while $\alpha = 2.5$ and $\alpha = 3.5$ produce too much high and low frequency content, respectively.

COMPARISON (Synthesizing sounds with different training data): We present examples of identical simulation scenarios in which we synthesize sounds using texture synthesis with different training audio clips. We see that we can obtain desirable variation in the sounds produced by changing the training data.

COMPARISON (With and without dynamic range mapping): We present an example in which texture synthesis has difficulty producing coherent output when the dynamic re-mapping procedure from §6.5.3 is omitted.

6.7 Conclusion

We presented a method for synthesizing plausible sounds synchronized with physically based simulations of fire. Our hybrid approach produces sounds from simulations time-stepped at low rates, then introduces high-frequency content as a post-process using either spectral bandwidth extension or texture synthesis techniques.

Limitations and Future Work: Our results are somewhat restricted by the limited fidelity of signals produced by the methods of §6.3. They tend to have a somewhat "bursty" character, and lack some of the mid-frequency "whooshing" behavior present in real flames. Recovering this behavior may require more sophisticated modeling of combustion chemistry in the underlying flame solver, which remains an open problem. An efficient computational model of turbulence and its impact on combustion sound is one potential area for future work. Alternately, it may be possible to produce plausible flame sounds without resorting to the costly fluid simulations used in this work. A sound

model that is compatible with low-frequency, low-resolution fluid simulation and can be effectively extended using texture synthesis is another area of potential future research.

The dynamic range mapping technique presented in §6.5.3 makes our texture synthesis more practical, but in some instances the method still has difficulty producing a suitable, temporally coherent output sound. This can occur in cases when the low-frequency input has a very wide dynamic range, while the training data has a small range, e.g., training samples that consist of steady rumbling sounds.

Physically principled spatialization of sounds generated by our method requires further investigation. The low-frequency sound model (§6.3) describes monopole sound sources distributed on a moving surface. In principle, this model could be used to produce spatialized multi-channel sound. Spatialization is less clear for sounds synthesized with bandwidth extension or texture synthesis.

CHAPTER 7

CONCLUSION

7.1 Summary and Conclusions

While automated sound synthesis for computer graphics has received a growing amount of attention over the last several years, many existing algorithms produce unsatisfying results for a wide variety of phenomena. The lack of methods for producing realistic soundtracks for computer animations severely limits our ability to produce compelling and immersive virtual environments.

In this thesis, we have begun to address these limitations by introducing principled sound synthesis algorithms for a variety of phenomena commonly modeled in computer animations and virtual environments. We began by introducing numerous new methods for improved rigid-body sound synthesis. Compared to prior methods, the approach presented in Chapter 3 allows us to produce significantly richer sounds for objects undergoing nonlinear vibrations. We further improved upon the rigid-body sound pipeline in Chapters 4 and 5, in which we provided efficient algorithms for synthesizing acceleration sound. This corrects a long-standing omission in rigid-body sound research, and allows us to produce more accurate and realistic sounds for a variety of multibody collision scenarios. Finally, in Chapter 6 we showed how hybrid methods based on both physics and real data can be used to synthesize sound for other, non-rigid phenomena such as combustion. Together, these contributions substantially expand the sound synthesis landscape, allowing users and researchers to produce more immersive results than those available with prior methods.

7.2 Limitations and Future Work

Potential avenues for future work building upon this thesis are manifold. The rigid-body sound algorithms presented in Chapters 3-5 all focus on the problem of producing sound from a single object. Although results are presented for multibody scenes, acoustic interactions between objects are not modeled. A great deal of research has been invested in to the problem of environmental acoustics, but this work has largely focused on modeling the acoustic response within largely static environments (e.g., classrooms, concert halls, etc.). Developing algorithms for resolving acoustic interactions between objects in scenes like the ones presented in our results remains a challenging open area for future research.

The modal sound algorithm combined with acceleration sound synthesis can efficiently and accurately synthesize sounds for a variety of small objects (see Chapters 4-5). We can also extend this algorithm to handle larger objects which exhibit nonlinear behavior (Chapter 3). Nevertheless, this method is still has fundamental limitations. The objects presented in our results are all relatively small (typically no more than 1m in size). Large objects tend to require significantly larger modal decompositions than the smaller objects presented in our results. Synthesizing sound with high mode counts negates the computational advantage of modal synthesis. Scalable algorithms are still required to efficiently synthesize sound from larger structures (e.g., buildings, construction equipment, etc.).

We also observe from the results in Chapters 3 and 4 that there appears to be a fundamental disconnect between numerical error and perceivable inaccuracies in sound synthesis results. For examples, FFAT maps from Chapter 3 with very low numerical accuracy appear to produce sound synthesis results nearly indistinguishable from re-

sults synthesized with far greater numerical precision. Perceptual studies of synthesized sounds are required to gain an understanding of the degree to which numerical accuracy is related to audible realism.

Finally, hybrid synthesis algorithms such as the one proposed in Chapter 6 may provide a promising alternative to purely physics-based approaches for phenomena other than fire (e.g., aerodynamic noise or cloth [121]). However, we must still address the problem of how to combine sounds produced by these algorithms with purely physics-based results in order to build virtual environments with a wide range of sound sources.

APPENDIX A

SCALING RELATIONSHIP PROOF

Following the notation of §5.2.1, consider object O with PAN fields $p_i^{(h)}$ and object O_β – a uniformly scaled version of object O. We first show that (6.2) holds for $P_i^{(\beta h)}$ on the domain Ω_β. Consider $\boldsymbol{x} \in \Omega_\beta, t \geq 0$ and let $\mathbf{y} = \boldsymbol{x}/\beta, \tau = t/\beta$. It is clear that if $\boldsymbol{x} \in \Omega_\beta$ then $\mathbf{y} \in \Omega$. For $i = 1, 2, 3$ we have $\frac{1}{c^2}\frac{\partial^2 P_i^{(\beta h)}(\boldsymbol{x},t)}{\partial t^2} = \frac{1}{\beta c^2}\frac{\partial^2 p_i^{(h)}(\mathbf{y},\tau)}{\partial \tau^2}$ by definition (5.2) and the chain rule. We also have $\nabla^2 P_i^{(\beta h)}(\boldsymbol{x}, t) = \frac{1}{\beta}\nabla^2_{\mathbf{y}} p_i^{(h)}(\mathbf{y}, t)$. Since $p_i^{(h)}$ satisfies (6.2) on Ω, it follows from these equations that $\frac{1}{c^2}\frac{\partial^2 P_i^{(\beta h)}(\boldsymbol{x},t)}{\partial t^2} = \nabla^2 P_i^{(\beta h)}(\boldsymbol{x}, t)$ on Ω_β, as required. The proof for $i = 4, 5, 6$ is analogous.

Next, we show that the boundary conditions (5.1) hold for $P_i^{(\beta h)}$. Let $\boldsymbol{x} \in \partial\Omega_\beta$ and define $\mathbf{y}$ and τ as before. Clearly $\mathbf{y} \in \partial\Omega$ and $\mathbf{n}_\beta(\boldsymbol{x}) = \mathbf{n}(\mathbf{y})$ where $\mathbf{n}$ and $\mathbf{n}_\beta$ are the normal fields on O and O_β, respectively. We will prove that (5.1) holds for $i = 4, 5, 6$ since the proof for $i = 1, 2, 3$ is similar. Following from (5.1) and (5.2), for $i = 4, 5, 6$, we have $\nabla P_i^{(\beta h)}(\boldsymbol{x}, t) \cdot \mathbf{n}(\boldsymbol{x}) = \beta \nabla_{\mathbf{y}} p_i^{(h)}(\mathbf{y}, \tau) \cdot \mathbf{n}(\mathbf{y})$ by the definition of $P_i^{(\beta h)}$ and the chain rule. We also have $\psi(t; \beta h) g_i(\boldsymbol{x}) = \psi(\beta\tau; \beta h) g_i(\boldsymbol{x}) = \beta\psi(\tau; h) g_i(\mathbf{y})$, which is true since $\psi(\beta\tau; \beta h) = \psi(\tau; h)$ and $g_i(\boldsymbol{x}) = -\rho\left(\mathbf{e}_{i-3} \times (\boldsymbol{x} - \boldsymbol{x}_0)\right) \cdot \mathbf{n}(\boldsymbol{x}) = -\beta\rho\left(\mathbf{e}_{i-3} \times (\mathbf{y} - \mathbf{y}_0)\right) \cdot \mathbf{n}(\mathbf{y})$. Therefore,

$$\begin{aligned}
\nabla P_i^{(\beta h)}(\boldsymbol{x}, t) \cdot \mathbf{n}(\boldsymbol{x}) &= \beta \nabla_y p_i^{(h)}(\mathbf{y}, \tau) \\
&= \beta \Psi(\tau; h) g_i(\mathbf{y}) \ \text{ since } p_i^{(h)} \text{ satisfies (5.1)} \\
&= \Psi(t; \beta h) g_i(\boldsymbol{x}),
\end{aligned}$$

as required. □

BIBLIOGRAPHY

[1] D. I. Abugov and O. I. Obrezkov. Acoustic noise in turbulent flames. *Combustion, Explosion, and Shock Waves*, 14:606–612, 1978.

[2] Jean-Marie Adrien. The missing link: Modal synthesis. In *Representations of musical signals*, pages 269–298. MIT Press, Cambridge, MA, USA, 1991.

[3] Steven S. An, Theodore Kim, and Doug L. James. Optimizing cubature for efficient integration of subspace deformations. *ACM Trans. Graph.*, 27(5):1–10, 2008.

[4] R. Annadana, H. EV, D. Sinha, and A. Ferreira. A Novel Audio Post-Processing Toolkit for the Enhancement of Audio Signals Coded at Low Bit Rates. In *AES 123rd Convention, New York, NY, USA, 2007 October 58*, 2007.

[5] Zhaosheng Bao, Jeong-Mo Hong, Joseph Teran, and Ronald Fedkiw. Fracturing Rigid Materials. *IEEE Transactions on Visualization and Computer Animation*, 13(2):370–378, 2007.

[6] David Baraff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. In *Computer Graphics (Proceedings of SIGGRAPH 89)*, pages 223–232, July 1989.

[7] David Baraff and Andrew P. Witkin. Large steps in cloth simulation. In *Proceedings of SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, pages 43–54, July 1998.

[8] Klaus-Jürgen Bathe. *Finite Element Procedures*. Prentice Hall, second edition, 1996.

[9] M. Bergou, M. Wardetzky, D. Harmon, D. Zorin, and E. Grinspun. A quadratic bending model for inextensible surfaces. In *Eurographics Symposium on Geometry Processing*, pages 227–230, 2006.

[10] S. Bilbao. A family of conservative finite difference schemes for the dynamical von Karman plate equations. *Numerical Methods for Partial Differential Equations*, 24(1), 2008.

[11] Javier Bonet and Richard D. Wood. *Nonlinear Continuum Mechanics for Finite Element Analysis*. Cambridge University Press, New York, second edition, 2008.

[12] Nicolas Bonneel, George Drettakis, Nicolas Tsingos, Isabelle Viaud-Delmon, and Doug James. Fast Modal Sounds with Scalable Frequency-Domain Synthesis. *ACM Transactions on Graphics*, 27(3):24:1–24:9, August 2008.

[13] R. Bridson, S. Marino, and R. Fedkiw. Simulation of clothing with folds and wrinkles. In *2003 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 28–36, August 2003.

[14] Robert Bridson, Jim Hourihim, and Marcus Nordenstam. Curl-noise for procedural fluid flow. *ACM Transactions on Graphics*, 26(3), July 2007.

[15] C. Phillip Brown and Richard O. Duda. A Structural Model for Binaural Sound Synthesis. *IEEE Trans. on Speech and Audio Processing*, 6(5), 1998.

[16] P. J. Burt and E. H. Adelson. A multiresolution spline with application to image mosaics. *ACM Transactions on Graphics*, 2(4):217–236, October 1983.

[17] M. Cardle, S. Brooks, Z. Bar-Joseph, and P. Robinson. Sound-by-numbers: motion-driven sound synthesis. In *2003 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 349–356, August 2003.

[18] Jeffrey N. Chadwick, Steven S. An, and Doug L. James. Harmonic Shells: A Practical Nonlinear Sound Model for Near-Rigid Thin Shells. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2009)*, 28(3), December 2009.

[19] Jeffrey N. Chadwick and Doug L. James. Animating fire with sound. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2011)*, 30(4), August 2011.

[20] Jeffrey N. Chadwick, Changxi Zheng, and Doug L. James. Faster acceleration noise for multibody animations using precomputed soundbanks. *ACM/Eurographics Symposium on Computer Animation*, 2012.

[21] Jeffrey N. Chadwick, Changxi Zheng, and Doug L. James. Precomputed acceleration noise for improved rigid-body sound. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2012)*, 31(4), August 2012.

[22] A. Chaigne, C. Touzé, and O. Thomas. Nonlinear vibrations and chaos in gongs and cymbals. *Acoustical Science and Technology*, 26(5):403–409, 2005.

[23] Antoine Chaigne and Christophe Lambourg. Time-domain simulation of damped impacted plates. i. theory and experiments. *Journal of the Acoustical Society of America*, 109(4):1422–1432, 2001.

[24] Anish Chandak, Christian Lauterbach, Micah Taylor, Zhimin Ren, and Dinesh Manocha. Ad-frustum: Adaptive frustum tracing for interactive sound propagation. 2008.

[25] D. Chapelle and K.J. Bathe. *The finite element analysis of shells*. Springer, 2003.

[26] M.G. Choi, S. Yong Woo, and H.S. Ko. Real-Time Simulation of Thin Shells. In *Computer Graphics Forum*, volume 26, pages 349–354. Blackwell Publishing Ltd, 2007.

[27] D. G. Chrighton, A. P. Dowling, J. E. Ffowcs Williams, M. Heckl, and F. G. Leppington. *Modern Methods in Analytical Acoustics*. Springer-Verlag, 1992.

[28] F. Cirak and M. Ortiz. Fully C^1-conforming subdivision elements for finite deformation thin-shell analysis. *Internat. J. Numer. Methods Engrg.*, 51:813–833, 2001.

[29] F. Cirak, M. Ortiz, and P. Schroder. Subdivision surfaces: A new paradigm for thin-shell finite-element analysis. *Internat. J. Numer. Methods Engrg.*, 47:2039–2072, 2000.

[30] Paul Clavin and Eric D. Siggia. Turbulent premixed flames and sound generation. *Combustion Science and Technology*, 78:147–155, 1991.

[31] Perry Cook. *Real Sound Synthesis for Interactive Applications*. A.K. Peters, 2002.

[32] L. Cremer, M. Heckl, and E.E. Ungar. *Structure Borne Sound : Structural Vibrations and Sound Radiation at Audio Frequencies*. Springer, 2nd edition, January 1990.

[33] Ingrid Daubechies. Orthonormal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics*, 41(7), 1988.

[34] W. Desmet. Mid-frequency vibro-acoustic modelling: challenges and potential solutions. In *Proceedings of ISMA 2002*, volume II, 2002.

[35] Y. Dobashi, T. Yamamoto, and T. Nishita. Real-time rendering of aerodynamic sound using sound textures based on computational fluid dynamics. *ACM Transactions on Graphics*, 22(3):539–545, July 2003.

[36] Y. Dobashi, T. Yamamoto, and T. Nishita. Synthesizing sound from turbulent fields using sound textures for interactive fluid simulation. *Computer Graphics Forum*, 23(3):736–744, September 2004.

[37] S. Dubnov, Z. Bar-Joseph, R. El-Yaniv, D. Lischinski, and M. Werman. Synthesizing sound textures through wavelet tree learning. *Computer Graphics and Applications, IEEE*, 22(4):38–48, 2002.

[38] A.A. Efros and T.K. Leung. Texture synthesis by non-parametric sampling. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1033–1038. IEEE, 1999.

[39] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 341–346, August 2001.

[40] M. Endo, S. Nishi, M. Nakagawa, and M. Sakata. Sound radiation from a circular cylinder subjected to elastic collision by a sphere. *Journal of Sound and Vibration*, 75(2):285–302, 1981.

[41] D. Enright, S. Marschner, and R. Fedkiw. Animation and rendering of complex water surfaces. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2002)*, 21(4):736–744, 2002.

[42] Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. Visual simulation of smoke. In Eugene Fiume, editor, *Proceedings of SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 15–22. ACM, ACM Press / ACM SIGGRAPH, 2001.

[43] Bryan E. Feldman, James F. O'Brien, and Okan Arikan. Animating suspended particle explosions. In *Proceedings of SIGGRAPH 2003*, pages 708–715, 2003.

[44] NH Fletcher. The nonlinear physics of musical instruments. *Reports on Progress in Physics*, 62(5):723–764, 1999.

[45] Paulo Flores, Jorge Ambrósio, José Carlos Pimenta Claro, and H. M. Lakarani. Influence of the contact-impact force model on the dynamic response of multi-body systems. In *Proceedings of the Institution of Mechanical Engineers. Part K: Journal of Multi-Body Dynamics*, volume 220, pages 21–34, 2006.

[46] Paulo Flores, Jorge A. C. Ambrosio, J. C. Pimenta Claro, and Hamid M.

Lankarani. chapter 3: Contact-Impact Force Models for Mechanical Systems. Springer, 2008.

[47] Paulo Flores, Margarida Machado, and Miguel T. Silva. On the continuous contact force models for soft materials in multibody dynamics. *Multibody System Dynamics*, 25(3):357–375, 2011.

[48] Nick Foster and Dimitris Metaxas. Modeling the motion of a hot, turbulent gas. In *Proc. of SIGGRAPH 1997*, pages 181–188, 1997.

[49] A. Frendi, L. Maestrello, and A. Bayliss. Coupling between plate vibration and acoustic radiation. *Journal of Sound and Vibration*, 177(2):207–226, 1994.

[50] Thomas Funkhouser, Nicolas Tsingos, Ingrid Carlbom, Gary Elko, Mohan Sondhi, James E. West, Goopal Pingali, and Addy Ngan. A beam tracing method for interactive architectural acoustics. *Journal of the Acoustical Society of America*, 115(3):739–756, February 2004.

[51] Thomas A. Funkhouser, Patrick Min, and Ingrid Carlbom. Real-time acoustic modeling for distributed virtual environments. In *Proceedings of ACM SIGGRAPH 1999*, Computer Graphics Proceedings, Annual Conference Series, pages 341–346, August 1999.

[52] A. Garg, E. Grinspun, M. Wardetzky, and D. Zorin. Cubic shells. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 91–98, 2007.

[53] G. Gilardi and I. Sharf. Literature survey of contact dynamics modelling. *Mechanism and Machine Theory*, 37(10):1213–1239, 2002.

[54] Yotam Gingold, Adrian Secord, Jefferson Y. Han, Eitan Grinspun, and Denis Zorin. A Discrete Model for Inelastic Deformation of Thin Shells. Technical report, Courant Institute of Mathematical Sciences, New York University, Aug 2004.

[55] Herbert Goldstein, Charles Poole, and John Safko. *Classical Mechanics*. Addison Wesley, San Francisco, California, third edition, 2002.

[56] S. Green, G. Turkiyyah, and D. Storti. Subdivision-based multilevel methods for large scale engineering of thin shells. In *Proceedings of ACM Solid Modeling*, pages 265–272, 2002.

[57] E. Grinspun, A. Hirani, M. Desbrun, and P. Schroder. Discrete shells. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 62–67, 2003.

[58] Eran Guendelman, Robert Bridson, and Ronald Fedkiw. Nonconvex rigid bodies with stacking. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2003)*, 22(3), August 2003.

[59] N. Gumerov and R. Duraiswami. *Fast Multipole Methods for the Helmholtz Equation in Three Dimensions*. Elsevier Science, 2005.

[60] R. W. Hamming. *Digital Filters*. Prentice-Hall, Englewood Cliffs, NJ, 1983.

[61] David J. Heeger and James R. Bergen. Pyramid-based texture analysis/synthesis. In *Proc. of SIGGRAPH 1995*, pages 229–238, 1995.

[62] H. Hertz. Über die Berührung fester elastiche Körper and über die harte (On the contact of elastic solids). *J. reine und angewandte Mathematk*, 92:156–171, 1882.

[63] Gerhard Hippmann. An algorithm for compliant contact between complexly shaped bodies. *Multibody System Dynamics*, 12(4):345–362, 2004.

[64] Jeong-Mo Hong, Tamar Shinar, and Ronald Fedkiw. Wrinkled flames and cellular patterns. *ACM Transactions on Graphics*, 26(3):47:1–47:6, July 2007.

[65] Christopher Horvath and Willi Geiger. Directable, High-Resolution Simulation of Fire on the GPU. *ACM Transactions on Graphics*, 28(3):41:1–41:8, July 2009.

[66] M. S. Howe. *Theory of Vortex Sound*. Cambridge University Press, 2003.

[67] Thomas J. R. Hughes. *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. Dover Publications, Inc., Mineola, New York, second edition, 2000.

[68] K. H. Hunt and F. R. E. Crossley. Coefficient of restitution interpreted as damping in vibroimpact. *Journal of Applied Mechanics*, 42(2):440–445, 1975.

[69] Matthias Ihme, Heinz Pitsch, and Daniel Bodony. Radiation of noise in turbulent non-premixed flames. In *Proceedings of the Combustion Institute*, volume 32, pages 1545–1553, 2009.

[70] Doug L. James, Jernej Barbič, and Dinesh K. Pai. Precomputed Acoustic Transfer: Output-sensitive, accurate sound generation for geometrically complex vibration sources. *ACM Transactions on Graphics*, 25(3):987–995, July 2006.

[71] Doug L. James and Dinesh K. Pai. DyRT: Dynamic response textures for real time deformation simulation with graphics hardware. *ACM Transactions on Graphics*, 21(3):582–585, July 2002.

[72] Doug L. James and Dinesh K. Pai. BD-Tree: Output-sensitive collision detection for reduced deformable models. *ACM Transactions on Graphics*, 23(3):393–398, August 2004.

[73] K. L. Johnson. *Contact Mechanics*. Cambridge University Press, 1985.

[74] Theodore Kim, Nils Thürey, Doug James, and Markus Gross. Wavelet turbulence for fluid simulation. *ACM Transactions on Graphics*, 27(3):50:1–50:6, August 2008.

[75] L. L. Koss and R. J. Alfredson. Transient sound radiated by spheres undergoing and elastic collision. *Journal of Sound and Vibration*, 27(1):59–75, 1973.

[76] P. Krysl, S. Lall, and J. E. Marsden. Dimensional model reduction in non-linear finite element dynamics of solids and structures. *International Journal for Numerical Methods in Engineering*, 51:479–504, 2001.

[77] Christophe Lambourg, Antoine Chaigne, and Denis Matignon. Time-domain simulation of damped impacted plates. ii. numerical model and results. *Journal of the Acoustical Society of America*, 109(4):1433–1447, 2001.

[78] H. M. Lankarani and P. E. Nikravesh. A contact force model with hysteresis damping for impact analysis of multibody systems. *Journal of Mechanical Design*, 112:369–376, 1990.

[79] E.R. Larsen and R.M. Aarts. *Audio bandwidth extension: application of psychoacoustics, signal processing and loudspeaker design*. Wiley, 2004.

[80] R. B. Lehoucq, D. C. Sorensen, and C. Yang. Arpack users guide: Solution of large scale eigenvalue problems by implicitly restarted arnoldi methods., 1997.

[81] C.M. Liu, W.C. Lee, and H.W. Hsu. High frequency reconstruction for band-limited audio signals. *Proc. DAFX-03, Sept*, 2003.

[82] Qing-Huo Liu and Kianping Tao. The perfectly matched layer for acoustic waves in absorptive media. *Journal of the Acoustical Society of America*, 102(4):2072–2082, 1997.

[83] Y. J. Liu. *Fast Multipole Boundary Element Method: Theory and Applications in Engineering*. Cambridge University Press, Cambridge, 2009.

[84] Brandon Lloyd, Nikunj Raghuvanshi, and Naga K. Govindaraju. Sound synthesis for impact sounds in video games. In *Proceedings of I3D 2011 Symposium on Interactive 3D Graphics and Games*, 2011.

[85] William E. Lorenson and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of SIGGRAPH 1987*, pages 163–169, 1987.

[86] P. Malatkar. *Nonlinear vibrations of cantilever beams and plates*. PhD thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA, 2003.

[87] Damian Marelli, Mitsuko Aramaki, Richard Kronland-Martinet, and Charles Verron. Time-frequency synthesis of noisy sounds with narrow spectral components. *IEEE Transactions on Audio, Speech and Language Processing*, 18(8):399–414, November 2010.

[88] MATLAB. *version 7.10.0 (R2010a)*. The MathWorks Inc., Natick, Massachusetts, 2010.

[89] J.H. McDermott, A.J. Oxenham, and E.P. Simoncelli. Sound texture synthesis via filter statistics. In *Applications of Signal Processing to Audio and Acoustics, 2009. WASPAA'09. IEEE Workshop on*, pages 297–300. IEEE, 2009.

[90] K. Mehraby, H. Khademhosseini, and M. Poursina. Impact Noise Radiated by Collision of Two Spheres: Comparison Between Numerical Simulations, Experiments and Analytical Results. *Journal of Mechanical Science and Technology*, 25(7):1675–1685, 2011.

[91] Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. *VisMath*, 2002.

[92] Don P. Mitchell and Arun N. Netravali. Reconstruction filters in computergraphics. In *Proceedings of SIGGRAPH 1988*, pages 221–228, 1988.

[93] William Moss, Hengchin Yeh, Jeong mo Hong, Ming C. Lin, and Dinesh

Manocha. Sounding liquids: Automatic sound synthesis from fluid simulation. *ACM Transactions on Graphics*, 29(3):21:1–21:13, June 2010.

[94] F. Moussaoui and R. Benamar. Non-linear vibrations of shell-type structures: a review with bibliography. *Journal of Sound and Vibration*, 255(1):161–184, 2002.

[95] Matthias Muller, David Charypar, and Markus Gross. Particle-based fluid simulation for interactive applications. In *ACM SIGGRAPH Symposium on Computer Animation (SCA)*, pages 154–159, 2003.

[96] A.H. Nayfeh and D.T. Mook. *Nonlinear oscillations*. Wiley-Interscience, 1979.

[97] AH Nayfeh and SA Nayfeh. Nonlinear normal modes of a continuous system with quadratic nonlinearities. *Journal of Vibration and Acoustics*, 117:199, 1995.

[98] Duc Quang Nguyen, Ronald Fedkiw, and Henrik Wann Jensen. Physically based modeling and animation of fire. *ACM Transactions on Graphics*, 21(3):721–728, July 2002.

[99] James F. O'Brien, Perry R. Cook, and Georg Essl. Synthesizing sounds from physically based motion. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 529–536, August 2001.

[100] James F. O'Brien, Chen Shen, and Christine M. Gatchalian. Synthesizing sounds from rigid-body simulations. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 175–181, July 2002.

[101] Stefano Papetti, Federico Avanzini, and Davide Rocchesso. Numerical methods for a nonlinear impact model: A comparative study with closed-form corrections. *IEEE Transactions on Audio, Speech and Language Processing*, 19(7):2146–2158, 2011.

[102] Alex Pentland and John Williams. Good vibrations: Modal dynamics for graphics and animation. In *Computer Graphics (Proceedings of SIGGRAPH 89)*, pages 215–222, July 1989.

[103] J.R. Pierce and S.A. Van Duyne. A passive nonlinear digital filter design which facilitates physics-based sound synthesis of highly nonlinear musical instruments. *The Journal of the Acoustical Society of America*, 101:1120, 1997.

[104] Nikunj Raghuvanshi and Ming C. Lin. Interactive Sound Synthesis for Large

Scale Environments. In *SI3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 101–108, New York, NY, USA, 2006. ACM Press.

[105] Nikunj Raghuvanshi, John Snyder, Ravish Mehra, Ming Lin, and Naga Govindaraju. Precomputed wave simulation for real-time sound propagation of dynamic sources in complex scenes. *ACM Transactions on Graphics (SIGGRAPH 2010)*, 29(3), 2010.

[106] Rajesh Rajaram and Tim Lieuwen. Acoustic radiation from turbulent premixed flames. *Journal of Fluid Mechanics*, 637:357–385, 2009.

[107] Zhimin Ren, Hengchin Yeh, and M.C. Lin. Synthesizing contact sounds between textured models. In *Virtual Reality Conference (VR), 2010 IEEE*, pages 139 –146, march 2010.

[108] Zhimin Ren, Hengchin Yeh, and Ming C. Lin. Example-guided physically based modal sound synthesis. *ACM Transactions on Graphics (to appear)*, 2013.

[109] E. J. Richards, M. E. Wescott, and R. K. Jayapalan. On the prediction of impact noise, i: Acceleration noise. *Journal of Sound and Vibration*, 62(4):547–575, 1979.

[110] E. J. Richards, M. E. Wescott, and R. K. Jayapalan. On the prediction of impact noise, ii: Ringing noise. *Journal of Sound and Vibration*, 65(3):419–451, 1979.

[111] C. Roads. *Microsound*. The MIT Press, 2004.

[112] Annie Ross and Germain Ostiguy. Propagation of the initial transient noise from an impacted plate. *Journal of Sound and Vibration*, 301(1):28–42, 2007.

[113] S. Schedin, C. Lambourge, and A. Chaigne. Transient sound fields from impacted plates: Comparison between numerical simulations and experiments. *Journal of Sound and Vibration*, 221(3):471–490, 1999.

[114] W. Schiehlen and R. Seifried. Three approaches for elastodynamic contact in multibody systems. *Multibody System Dynamics*, 12(1):1–16, 2004.

[115] Anna Schwarz and Johannes Janicka, editors. *Combustion Noise*. Springer, 2009.

[116] Xavier Serra and Julius Smith. Spectral modeling synthesis: A sound anal-

ysis/synthesis system based on a deterministic plus stochastic decomposition. *Computer Music Journal*, 14(4):12–24, 1990.

[117] Ahmed Shabana. *Dynamics of Multibody Systems*. Cambridge, 3rd edition, 2005.

[118] Ahmed A. Shabana. *Theory of Vibration, Volume II: Discrete and Continuous Systems*. Springer–Verlag, New York, NY, 1990.

[119] L. Shen and Y. J. Liu. An adaptive fast multipole boundary element method for three-dimensional acoustic wave problems based on the Burton-Miller formulation. *Computational Mechanics*, 40(3):461–472, 2007.

[120] Jos Stam. Stable fluids. In *Proceedings of SIGGRAPH 1999*, pages 121–128, August 1999.

[121] Doug L. James Steven S. An and Steve Marschner. Motion-driven concatenative synthesis of cloth sounds. *ACM Transactions on Graphics (SIGGRAPH 2012)*, August 2012.

[122] Christophe N. J. Stoelinga and Robert A. Lutfi. Modeling manner of contact in the synthesis of impact sounds for perceptual research. *Journal of the Acoustical Society of America*, 130(2):EL62–EL68, 2011.

[123] Warren C. Strahle. Some results in combustion generated noise. *Journal of Sound and Vibration*, 43(1):113–125, 1972.

[124] G. Strobl, G. Eckel, D. Rocchesso, and S. le Grazie. Sound texture modeling: A survey. In *Proceedings of the 2006 Sound and Music Computing (SMC) International Conference*, pages 61–5, 2006.

[125] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically Deformable Models. In *Computer Graphics (Proceedings of SIGGRAPH 87)*, pages 205–214, July 1987.

[126] Demetri Terzopoulos and Andrew Witkin. Physically Based Models with Rigid and Deformable Components. *IEEE Computer Graphics & Applications*, 8(6):41–51, November 1988.

[127] C. Touzé, O. Thomas, and A. Chaigne. Hardening/softening behaviour in non-linear oscillations of structural systems using non-linear normal modes. *Journal of Sound and Vibration*, 273(1-2):77–101, 2004.

[128] K. van den Doel and D. K. Pai. Synthesis of shape dependent sounds with physical modeling. In *Intl Conf. on Auditory Display*, Xerox PARC, Palo Alto, November 1996.

[129] Kees van den Doel, Paul G. Kry, and Dinesh K. Pai. FoleyAutomatic: Physically Based Sound Effects for Interactive Simulation and Animation. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 537–544, August 2001.

[130] Tony S. Verma and Tersa H. Y. Meng. Extending spectral modeling synthesis with transient modeling synthesis. *Computer Music Journal*, 24(2):47–59, 2000.

[131] M. Vorlander. *Auralization: Fundamentals of Acoustics, Modelling, Simulation, Algorithms and Acoustic Virtual Reality*. Springer Verlag, 2007.

[132] A. O. Wåhlin, P. O. Gren, and N-E. Molin. On structure borne sound: Experiments showing the initial transient acoustic wave field generated by an impacted plate. *Journal of the Acoustical Society of America*, 96(5):2791–2797, 1994.

[133] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of SIGGRAPH 2000*, pages 479–488, 2000.

[134] S.F. Wu. Methods for reconstructing acoustic quantities based on acoustic pressure measurements. *The Journal of the Acoustical Society of America*, 124(5):2680, 2008.

[135] Wang Yufang and Tong Zhongfang. Sound Radiated from the Impact of Two Cylinders. *Journal of Sound and Vibration*, 159(2):295–303, 1992.

[136] Changxi Zheng and Doug L. James. Harmonic fluids. *ACM Transactions on Graphics*, 28(3), August 2009.

[137] Changxi Zheng and Doug L. James. Rigid-body fracture sound with precomputed soundbanks. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2010)*, 29(3), July 2010.

[138] Changxi Zheng and Doug L. James. Toward high-quality modal contact sound. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2011)*, 30(4), August 2011.

[139] U. Zölzer and X. Amatriain. *DAFX: Digital Audio Effects*. John Wiley & Sons Inc, 2002.